Java • C++ • Perl • Forth

# Dr. Dobb's JOURNAL

#262 FEBRUARY 1997

SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

## SOFTWARE DEBUGGING & TESTING

- Regression Testing to Reduce Your Bug Counts
- Testing Java GUIs
- Windows NT Filesystem
- Testing Test Tools
- Scott Meyers on C++ Program Analyzers

PROGRAMMING MOBILE ROBOTS

ACCESSING SQL DATABASES FROM JAVA APPLETS

I-NODES AND THE LINUX FILESYSTEM

*AL STEVENS* STRESS TESTS THE MIDI DATA STREAM

$3.95 ($4.95 CANADA)

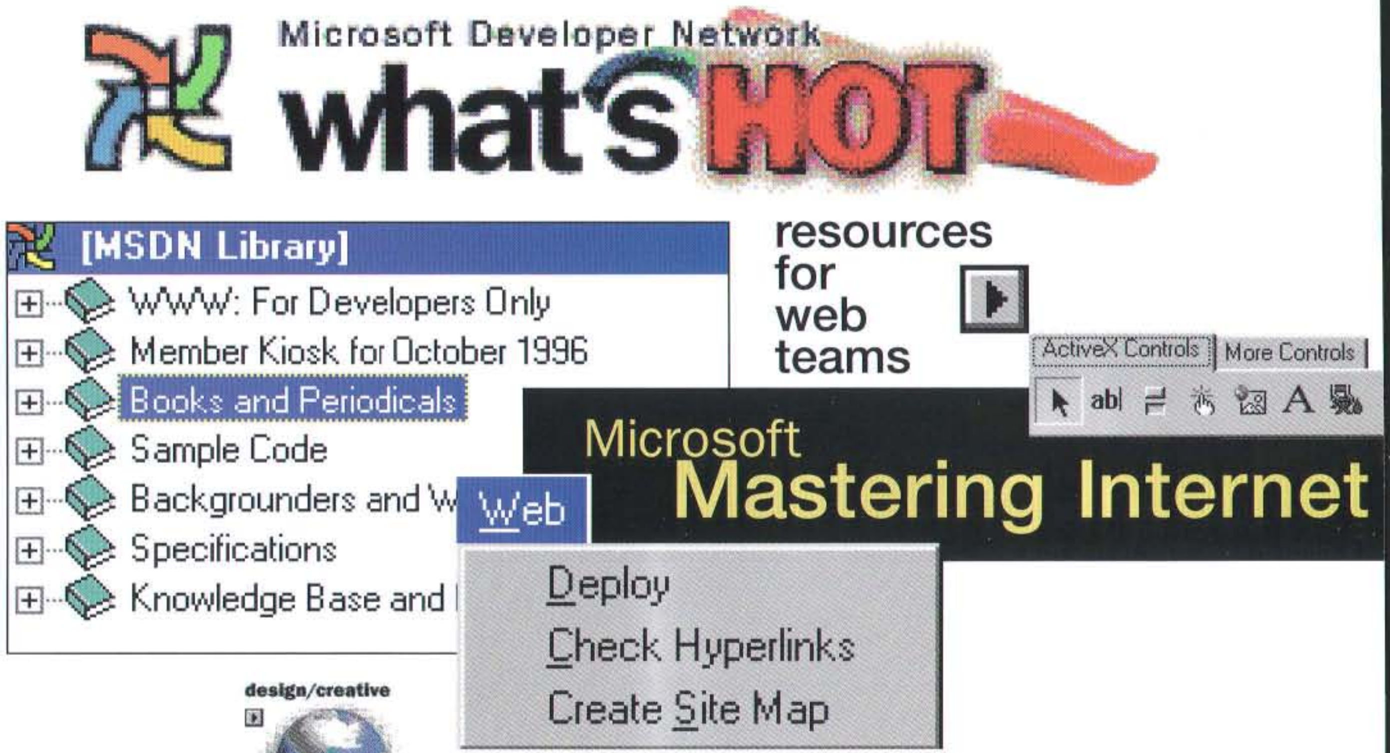
0 70992 35566 2 02>

A Miller Freeman Publication

# Microsoft Visual Tools.

the first official sup

for



[MSDN Library]

- WWW: For Developers Only
- Member Kiosk for October 1996
- Books and Periodicals
- Sample Code
- Backgrounders and W
- Specifications
- Knowledge Base and

**The Internet is creating plenty of excitement among developers these days. And plenty of extra work, too. With Microsoft's Visual Tools you can build on what you already know and count on the support you've come to expect from Microsoft.® We've focused on Internet standards, so you know what you build today will work tomorrow. For instance, with Microsoft Visual J++™ development software, you get Java™ language within a**

Think of them as port group Internet developers.

Development
web gallery
Microsoft Developer Studio
File Edit View Insert Build Tools Window Help
Wire Frame
Ready
Microsoft Visual J++ Professional Edition
Visual FoxPro
Developer Network
Visual SourceSafe
Mastering Internet Development
Visual C++
Microsoft Visual Basic Professional Edition

Site Builder network

proven, full-featured environment. Create your own ActiveX™ controls with Visual C++® development system to add Internet functionality. With Visual Basic® Scripting Edition built right in to Internet Explorer 3.0, you can automate objects and activate Web pages faster. Use Microsoft Visual SourceSafe™ version control system to coordinate team Website development and maintenance. As you'd expect, we'll help keep you current with the latest technologies. Visit www.microsoft.com/workshop/ for hundreds of lines of sample code, free controls, and information. Stay up to speed with the Microsoft Developer Network and learn with the Mastering Internet Development interactive training CD. Join the SiteBuilder Network and get even more support. Remember, you're not in this alone.

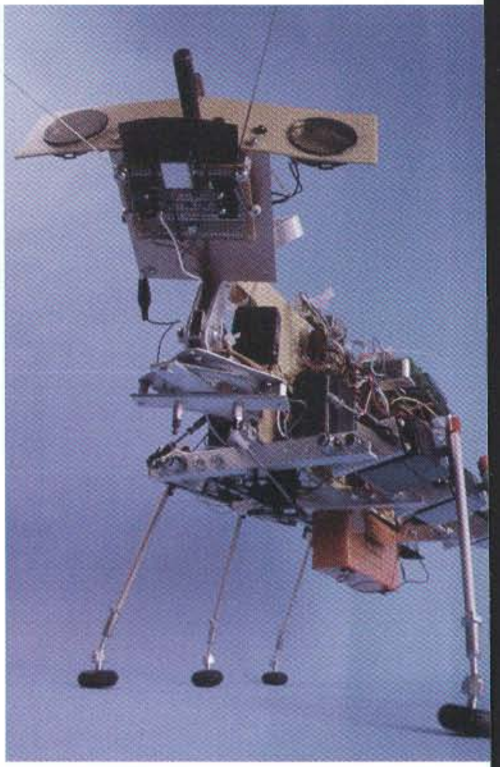
Microsoft®

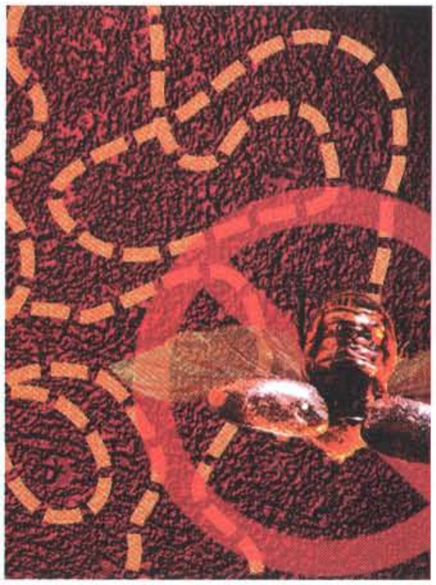
Where do you want to go today?™ www.microsoft.com/devonly/

Dr. Dobb's JOURNAL SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

# CONT

## FEATURES


*Cover photograph by Michael Carr*

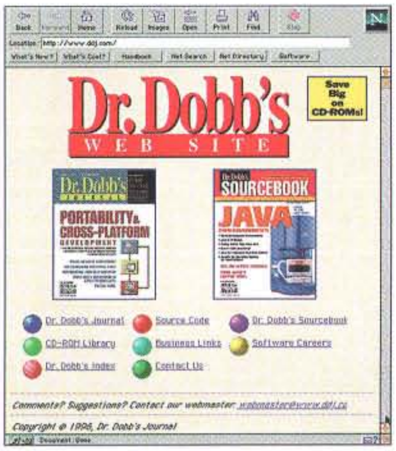


## EMBEDDED SYSTEMS

## NETWORKED SYSTEMS

ENTS

FEBRUARY 1997
VOLUME 22, ISSUE 2



## SOURCE CODE AVAILABILITY

As a service to our readers, all source code is available on a single disk and online. To order the disk, send $14.95 (California residents add sales tax) to *Dr. Dobb's Journal*, 411 Borel Ave., San Mateo, CA 94402, call 415-655-4100 x5701, or use your credit card to order by fax, 415-358-9749. Specify issue number and date. Code is also available through the DDJ Forum on CompuServe (type GO DDJ), via anonymous FTP from site ftp.mv.com (192.80.84.3) in the /pub/ddj directory, on the World Wide Web at http://www.ddj.com, and through DDJ Online, a free service accessible via direct dial at 415-358-8857 (14.4 kbps, 8-N-1).

## NEXT MONTH

Once shunned as AI off-shoots, intelligent agents have found new life on the World Wide Web. In March, we'll look at the issues and technologies behind mobile agents.

PLATFORMS: WINDOWS NT • WINDOWS 95 • OS/2 • NLM •

DOS • APPLE A/UX • LYNX • SCO • SYSTEM MANAGER • SUN SPARC-SOLARIS • IBM RS/6000 • HP9000 • RS/6000

• SUN O/S 4.X • SUN O/S 5.X • MIPS ABI (SGI) •

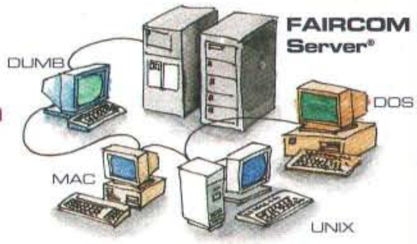
APPLE SYSTEM 7 • AT&T UNIX • BANYAN • INTERACTIVE UNIX • LINUX • MOTOROLA 88OPEN • QNX •

## Q: What does it take to deploy a superior client/server application? A: *A SUPERIOR SERVER*

***START*** **with the most advanced client-side SDK on the market: c-tree® Plus at $895.**
- Complete "C" Source code
- ROYALTY FREE (Client Side)
- Multiple supported protocols
- Fast, portable, reliable
- Powerful features like transaction processing
- Win95, NT, and Windows 3.1 ready

***RESULT?*** **A solid, economical, easily deployable product that fits your needs.**
- Portable
- Scalable
- Exceptional Performance
- Flexible
- Easy Server distribution
- Convenient OEM terms

***ADD*** **a strong, multi-platform, industrial-strength Server that supports.**
- File mirroring
- Heterogeneous networking
- Automatic disaster recovery
- Multi-threaded design
- Best price/performance available: from $445- $3745



Heterogeneous TCP/IP Network

**You can't find a better client SDK with these features!**
**Over sixteen years of proven reliability and performance.**
**No one else supports over 30 platforms in this price range!**

## c-tree Plus®
- Complete C Source
- Single/Multi User
- Client/Server (optional)
- Full ISAM functionality
- No Royalties
- Transaction Processing
- Fixed/Variable Length Records
- High Speed Data/Index Caching
- Batch Operations
- File Mirroring
- Multiple Contexts
- Unsurpassed Portability

## FairCom Server®
- Client/Server Model
- Transaction Processing
- Requires <2MB RAM
- Online Backup
- Disaster Recovery
- Rollback - Forward
- Anti-Deadlock Resolution
- Client-side "C" Source
- Multi-threading
- Heterogeneous networking
- File Mirroring
- OEM/Source Available

CIRCLE NO. 84 ON READER SERVICE CARD

**FOR YOUR NEXT PROJECT CALL FAIRCOM: YOU CAN'T FIND A BETTER HETEROGENEOUS CLIENT/SERVER SOLUTION!**

Also inquire about these FairCom products:

d-tree™ r-tree© ODBC Driver

FAIRCOM® CORPORATION

WWWeb Address: http://www.faircom.com/

**800-234-8180**

**U.S.A.** 4006 W. Broadway - Columbia, MO 65203-0100
phone (573) 445-6833 fax (573) 445-9698
**EUROPE** Via Patrioti, 6-24021 Albino (BG) - ITALY
phone (035) 773-464 fax (035) 773-806
**JAPAN** IKEDA Bldg. #3,4f-112-5, Komei-chou - Tsu-city,MIE 514 Japan
phone (0592) 29-7504 fax (0592) 24-9723

Dr. Dobb's JOURNAL — SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PUBLISHER *Peter Westerman*

EDITORIAL
EDITOR-IN-CHIEF *Jonathan Erickson*
MANAGING EDITOR *Kevin Carlson*
TECHNICAL EDITOR *Tim Kientzle*
ASSOCIATE TECHNICAL EDITOR *Eugene Eric Kim*
SENIOR PRODUCTION EDITOR *Monica E. Berg*
ASSOCIATE MANAGING EDITOR *Deirdre Blake*
EDITORIAL ASSISTANT *Amy Wu*
COPY EDITOR *Maureen Nelson*
ART DIRECTOR *Bruce Gardner*
CONTRIBUTING EDITORS *Al Stevens, Tom Genereaux, Scot Wingo, George Shepherd, Robert Collins, Cliff Berg, Steve Alexander, David Betz, Bruce Schneier, Mark Russinovich, Bryce Cogswell*
EDITOR-AT-LARGE *Michael Swaine*
GROUP PUBLISHER *Peter Hutchinson*
PRODUCTION MANAGER *Lisa Mancin*

CIRCULATION
DIRECTOR OF CIRCULATION *Jerry Okabe*
CIRCULATION MANAGER *Michael Poplardo*

MARKETING/ADVERTISING
ADVERTISING DIRECTOR *Charles P. Shively*
BUSINESS MANAGER *Jennifer Christen*
DIRECTOR OF MARKETING *Valerie Dow*
DIRECTOR OF ELECTRONIC PUBLISHING *Stan Barnes*
SALES AND MARKETING MANAGER *Holly Vessichelli*
ACCOUNT MANAGERS see page 121

MILLER FREEMAN INC.
CHAIRMAN/CEO *Marshall W. Freeman*
SENIOR VICE PRESIDENT/CFO *Warren "Andy" Ambrose*
SENIOR VICE PRESIDENTS *David Nussbaum, Darrell Denny, Donald A. Pazour, Wini D. Ragus*
SENIOR VICE PRESIDENT/SOFTWARE *Regina Starr Ridley*
VICE PRESIDENT/PRODUCTION *Andrew A. Mickus*
VICE PRESIDENT/CIRCULATION *Jerry Okabe*

***DR. DOBB'S JOURNAL* ISSN 1044-789X** is published monthly by Miller Freeman, Inc., 600 Harrison Street, San Francisco, CA 94107; 415-905-2200. Periodicals Postage Paid at San Francisco and at additional mailing offices.

**ARTICLE SUBMISSIONS:** Send manuscript and disk (with article, listings, and letter to the editor) to *DDJ* Submissions, 411 Borel Ave., San Mateo, CA 94402-3522.

***DDJ* ON COMPUSERVE:** Type GO DDJ.

***DDJ* ON THE INTERNET:** editors@ddj.com or http://www.ddj.com

**SUBSCRIPTION:** $34.95 for 1 year; $69.90 for 2 years. International orders must be prepaid. Payment may be made via Mastercard, Visa, or American Express; or via U.S. funds drawn on a U.S. bank. Canada and Mexico: $45.00 per year. All other foreign: $70.00 per year.

**FOR SUBSCRIPTION QUESTIONS,** change of address, and orders, call toll-free **800-456-1215** (U.S. and Canada). For all other countries, call **303-678-0439** or fax **303-661-1885**. On CompuServe, the customer-service number is 71572,341. Or write, *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188.

**BACK ISSUES** may be purchased for $9.00 per copy (includes shipping and handling). For issue availability, call **800-444-4881** in the U.S. and Canada. All other countries call **913-838-7500** or fax **913-841-2624**. Send e-mail to **orders@mfi.com.** Back issue orders must be prepaid. Please send payment to *Dr. Dobb's Journal*, 1601 W. 23rd Street, Suite 200, Lawrence, KS 66046-2700.

**POSTMASTER:** Send address changes to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80328-6188. **ISSN 1044-789X. GST (Canada) # R124771239.**

Canada Post International Publications Mail Product (Canadian Distribution) **Sales Agreement No. 0548677.**

**FOREIGN NEWSSTAND DISTRIBUTOR:** Worldwide Media Service Inc., 30 Montgomery St., Jersey City , NJ 07302; 212-332-7100.

Standard mail enclosed in the following versions: Ver. 6 and Ver. 7.

Estimated print run 170,000.

BULK RATE U.S. POSTAGE PAID WASECA, MN PERMIT NO. 218

Miller Freeman
A United News & Media publication

ABP American Business Press
Printed in the USA
BPA

# Learn Why NuMega BoundsChecker Is As Hot As JAVA and VRML!

JAVA and VRML have taken the internet market by storm with cutting edge development tools. But, BoundsChecker™ 4.0 for Windows NT® emerged as the 1996 PC Magazine Technical Excellence Award winner. Not hard to believe when over 80,000 developers already know that programming applications — internet or otherwise — require the best tool available. When your software quality is on the line, you need a proven dependable error detection solution. Make sure you use the best development tool for error detection. Use award-winning BoundsChecker.

## BoundsChecker catches the most errors.

BoundsChecker is the award-winning Advanced Error Detection™ tool for Windows. It catches API and OLE errors, Windows OS specific errors and C/C++ language errors — plus all of your static, stack and heap errors. BoundsChecker is ideal for component-based development.

## Smart Debugging™ means you spend less time debugging.

Its integration with the Visual C++ IDE enables powerful, transparent error detection as you do your normal debugging. There is no extra work on your part...errors are found...the exact location pinpointed... and a correction suggested.

## It's a Windows expert so you don't have to be.

BoundsChecker performs active API validation on more than 3,000 Windows APIs and OLE interfaces, including Win32®, ActiveX™, DirectX™, ODBC and many more.

## Download a Free Evaluation Copy at

www.numega.com

or Call 1-800-4-NUMEGA (800-468-6342) to order

**NuMega BoundsChecker™ 4**

**Advanced Error Detection™ for Windows®**

TECHNICAL EXCELLENCE
PC MAGAZINE
1996
THE NUMEGA BOUNDSCHECKER 4.0 DEVELOPMENT TEAM
BOUNDSCHECKER 4.0 PROFESSIONAL EDITION FOR WINDOWS NT
NUMEGA TECHNOLOGIES, INC.
DEVELOPMENT TOOLS CATEGORY

PC Magazine Editors' Choice · Visual Basic® Programer's Journal '96 Award Readers' Choice · Software Development Jolt Productivity Award 1995 · Windows Tech Journal Star Tech 1995 Product Excellence

9 Townsend West • Nashua, NH 03063 • 603 889-2386 • 800 4-NUMEGA
Fax 603 889-1135 • info@numega.com • www.numega.com

NuMega™ Technologies
Making Software Work

CIRCLE NO. 342 ON READER SERVICE CARD

# Power Plays

The high-tech electricity meters recently installed hereabouts continue to undergo testing. In replacing conventional meters, the utility company went house-to-house, plugging in new ones capable of transmitting and receiving data wirelessly. Like buggy whips and automobile fins, human meter readers are dust in the wind, at least in this burg. (Darn, there goes another career option.) Now, meters all over town fire up once a month, letting the central billing office know if Tom Bodett and the local Motel 6 really did leave a light on.

All in all, the local electrical provider installed more than 36,000 new meters that can transmit/receive data at 900 MHz within a range of 750 feet. Communicating with the meters are 700 intermediate antennas attached to streetlight poles around town. In turn, these intermediate devices communicate at 1.4 GHz with three primary antennas linked via fiber-optic land lines to the billing office about 30 miles away. It is the intermediate antennas that are (literally) driving engineers batty.

While there aren't a lot of tall buildings and high hills in town, there are plenty of tall trees to interfere with low-power radio transmissions. Consequently, utility engineers are having to test and retest to find the optimum layout for the intermediate antenna grid. Not wanting to miss a billing cycle in the meantime, "meter readers" are slowly driving a van—loaded with portable computers—up and down the city streets, activating meters and collecting data. At least in this case, mobile computing is just that.

Since big cities such as Denver, Pittsburgh, Spokane, and others already have similar systems up and running, it's a safe bet that the challenges in this small town aren't unique and will be resolved. As in those big cities, the system here is the Genesis Fixed Network automatic meter-reading system from Itron (Spokane, Washington). In addition to standard meter-reading capabilities, the system can provide daily readings, real-time on-demand readings, start-and-end of service readings, time-of-use readings, outage detection, and the ability to monitor inactive accounts for unauthorized usage. The system also can determine peak/off-hour power patterns; the utility companies may start billing us for electricity like phone companies do for long-distance phone service.

There's a power play of a different sort going on in Washington, D.C., at least in terms of software patents and copyrights. In particular, proposals in Congress would move the copyright office out of the Library of Congress and into the executive branch, specifically under the auspices of the Commerce Department's U.S. Patent and Trademark Office. The result would be a new "office of intellectual property policy." Behind the shift is the Commerce Department's desire for more control over intellectual property when setting international trade policy and negotiating trade deals. Less clear is how this change would actually benefit content owners.

This bureaucratic move is confounding because, as Joseph Yang points out in his excellent paper "Patenting Content: The Expanding Role of Patent Protection for Internet-based Information Products" (http://www.alumni.caltech.edu/~yang/), the distinctions between copyright and patents are blurring, in part due to executable content (aka "applets"). Historically, says Yang, who is both a patent attorney and an engineer, patent protection has been for functional aspects of a computer program, while copyright has protected its expressive elements. Today, however, the trend is toward merging copyright and patent concepts and enlarging the scope of patent protection for software. This is the stance put forward in the PTO's new "Examination Guidelines for Computer-Related Inventions."

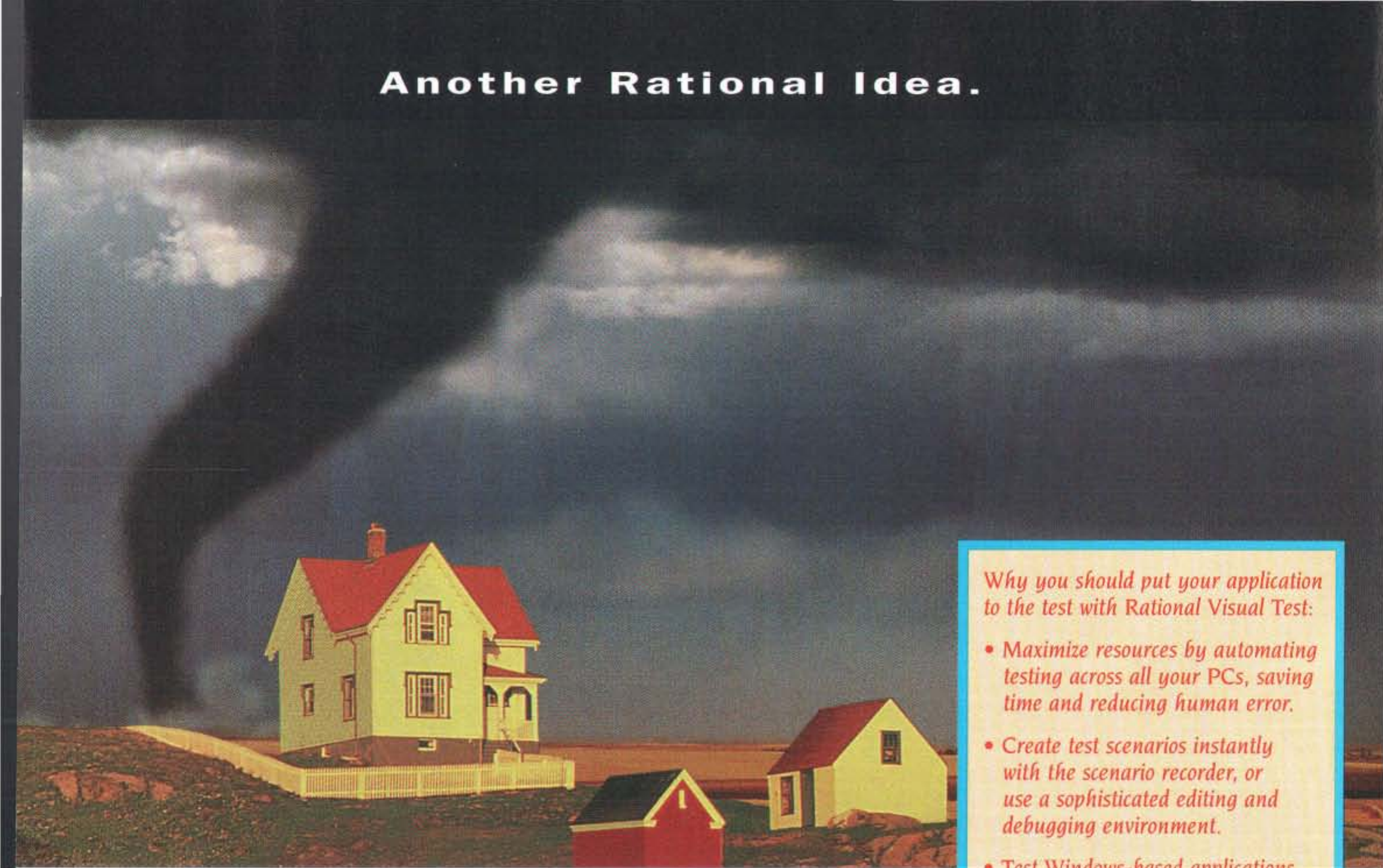
Especially during these periods of transition, patents continue to be a topic of vital importance to software developers. According to Greg Aharonian and his Internet Patent News Service, two significant trends related to software patents evolved in the first half of 1996: 919 network-software patents were issued (almost double the second-place count of 528 image-processing patents), and Microsoft moved rapidly into the software-patenting ranks. In 1995, Microsoft ranked 24th in the number of patents issued; in the first half of 1996, it climbed to 12th. When everything is totaled up, Aharonian predicts that 8100 new software patents will have been issued in 1996—almost as many as were issued in 1992, 1993, and 1994 combined.

In other words, in the four-year period between 1993 and 1996, approximately 20,000 software patents will have been issued. Now ask yourself if you really believe 20,000 novel innovations have been developed in the software world during that period. Yes, changes are needed in the copyright and patent arena, but the proposals on the table aren't intended to address the real problems.

Jonathan Erickson

Jonathan Erickson
editor-in-chief

**Another Rational Idea.**

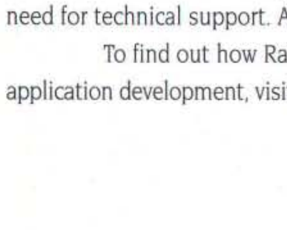


Why you should put your application to the test with Rational Visual Test:

- *Maximize resources by automating testing across all your PCs, saving time and reducing human error.*
- *Create test scenarios instantly with the scenario recorder, or use a sophisticated editing and debugging environment.*
- *Test Windows-based applications written in any language, and log the results to your database of choice.*
- *Get the power to test 32-bit applications, ActiveX controls and DLL files.*
- *Utilize the Advanced Error Detection™ system from NuMega™ BoundsChecker™*

## Always test your architecture as though the worst is yet to come.

**Presenting Rational Visual Test, the industry's easiest way to integrate automated testing into the development process.** There are plenty of good reasons to test an application prior to implementation. No one wants to deploy a less than top-quality product. But how can you prevent failures both now *and* in the future, as you make modifications two, three or even five years down the road?



The answer lies in automated testing during the development process. After all, software quality is not only the result of well-planned design, but also the elimination of random errors. Rational Visual Test allows you to retest your application automatically at every stage of development. Unattended, across every PC in your operation. Which means you discover errors early in the process and deliver better software in less time.

Of course, higher software quality contributes more than shorter development time. Fewer maintenance upgrades cut shipping and marketing costs. Higher profit margins result from a reduced need for technical support. And greater customer satisfaction tends to have an obvious effect on your bottom line.

To find out how Rational Visual Test (originally sold as Microsoft Visual Test) can enhance your application development, visit our web site (**www.rational.com/vtest**) or call **1-800-728-1212, ext. 214** today.

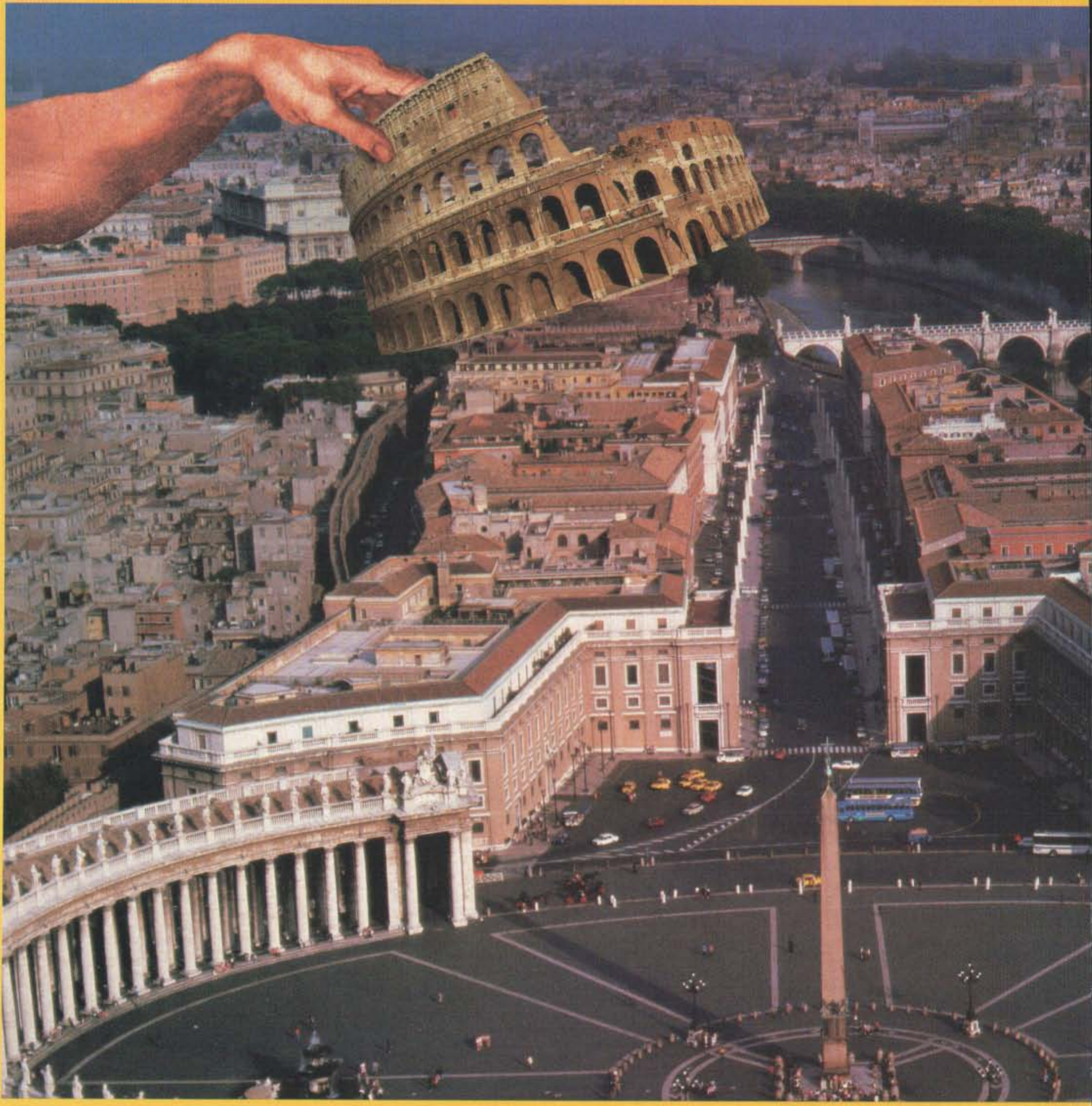
Designed for Microsoft® Windows®95

Microsoft Enterprise Development Partner

RATIONAL
SOFTWARE CORPORATION

CIRCLE NO. 104 ON READER SERVICE CARD

**Build Rome in a day.**
**It's easy with Symantec Visual Café™ for Java™.**



*Offer expires Feb. 28, 1997. Symantec is a registered trademark and Visual Café is a trademark of Symantec Corporation. Java is a trademark of Sun Microsystems, Inc. Windows is a registered trademark of Microsoft Corporation. Macintosh is a registered trademark of Apple Computer, Inc. Netscape Navigator is a trademark of Netscape Corporation. All other brand names or trademarks are property of their respective owners. ©1996 Symantec Corporation. All rights reserved. In Canada, call 1-800-365-8641. In Australia, call 02-879-6577. In Europe, call 31-71-535-3111.

**CIRCLE NO. 195 ON READER SERVICE CARD**

## Visual Java Development

Develop faster. Compile faster. Debug faster. Realize your Java™ dreams faster.

Symantec, the creator of the first full-featured Java development environment now unleashes the first Rapid Application Development environment for Java developers:

**Symantec Visual Café.™**

Visual Café comes complete with an extensible component library with all of the building blocks for your application.

Simply drag and drop a component onto a form. Our Interaction Wizard lets you visually specify the actions and events. And then Visual Café automatically generates the Java code.

And thanks to our exclusive two-way programming you can add or modify the code at the source level, too.

So, you'll be whipping out prototypes at speeds you can only dream of now.

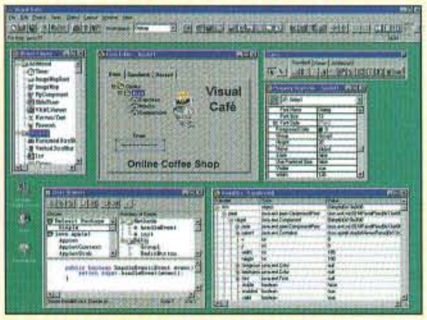
Imagine building all your forms visually. Or building your user interface without writing one single line of code! ▶

As an added bonus, Symantec's Just In Time compiler (JIT) (included in Netscape's upcoming Navigator™) runs your Java applications faster than any other browser or Java virtual machine on the planet.

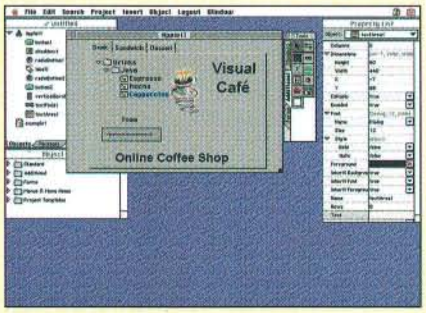
So get your hands on the hottest new development tool for Java now.

**Buy Visual Café today for just $199* and receive a $40.00 mail-in rebate. Purchase your copy online at cafe.symantec.com. Or pick up a copy at the MicroCenter, Fry's, CompUSA or Egghead nearest you. For more information, please call 1-800-453-1059 ext. 9A31.**

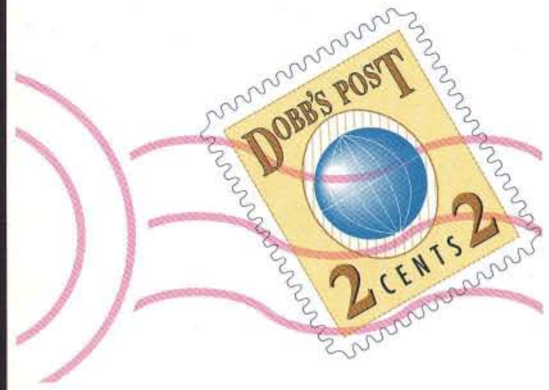
**Visual Tools for Windows®**



**Visual Tools for Macintosh®**



SYMANTEC.®



***Small C Appeal***

Dear *DDJ*,

I'm appealing to your readers for help in finding a copy of the book *A Small C Compiler* by James E. Hendrix. The publisher put it out of print. If anyone has a copy they're willing to sell, please contact me.

Martin Schaaf
P.O. Box 761
Alameda, CA 94501

*DDJ* responds: Martin, in response to requests like yours, we're about ready to release *Dr. Dobb's Small C Compiler Resource CD-ROM* that will include the full text to Jim's book *A Small C Compiler: Language, Usage, Theory, and Design*, collected *DDJ* articles on Small C, several implementations of the compiler, and related Small C tools.

***Expanding Retail Channels***

Dear *DDJ*,

I enjoyed reading Jonathan Erickson's November 1996 editorial and wholeheartedly agree with his assessment of how difficult it can be for software publishers to get placement in the retail channel. Our company, Online Interactive, offers an alternative solution for software publishers trying to distribute their products. Online Interactive operates a series of electronic download stores called the "atOnce Software" stores (http://www.atonce.com). Our first store opened on the Internet in January, 1996 and was one of the original participants in Microsoft's electronic software-distribution pilot program. Since then, we have opened additional stores on America Online, CompuServe, and the Microsoft Network. We have licensed over 125 software publishers, including GT Interactive, Intuit, and Asymetrix; more importantly, we represent a host of smaller publishers that are successfully using the download channel to generate revenue and reach customers. With over 1500 titles available for instant delivery, we will soon surpass the selection of even the largest computer superstores.

Robert Nachbar
RobertN@online-interactive.com

***Goal-Directed Design***

Dear *DDJ*,

Alan Cooper's September 1996 article "Goal-Directed Software Design" presents laudable principles. However, I'm always concerned when realistic examples don't derive from principles.

What really caught my eye was the caption for Figure 4: "The monthly view in Schedule+ is a paragon of pixels sacrificed on the altar of meaningless regularity." (The figure itself shows a traditional paper-style monthly calendar layout displayed on a computer screen with lots of wasted space.)

My technophile instinct wanted to jump up and applaud Cooper for advocating a move away from now-irrelevant limitations like the static nature of information presented on sheets of dead trees. And I certainly wouldn't choose to defend a Microsoft product under ordinary circumstances. Unfortunately, a few seconds later I realized there was a problem.

For in-house corporate developers, it might be possible to provide a more dynamic display or at least one which wastes less screen real estate. In this situation, users are haunted by the twin specters of necessity and inflexibility: The software is more likely to be mission-critical and less likely to be provided by more than one vendor. Users here are more likely to be willing to slog through the (admittedly small) paradigm shift because they've been shifting this way for years to accommodate alien software.

But for independent software vendors who must induce skeptical and often deeply conservative customers to part with hard-won cash, imposing a new calendar display paradigm is no way to win mindshare. These users are more likely to greet unfamiliar displays of familiar data with knee-jerk rejection or MEGO ("my eyes glaze over"). According to Cooper, this might be the user's fault; Cooper might say the user has a "false goal." But educating users is not usually the job of a software interface designer.

Of course, it's possible to satisfy all users by providing both a traditional and a more efficient calendar display. However, it's not always practical to develop both.

The principles of goal-directed user interface design sound good, but I'm still waiting for a "killer" case history.

Pete Gontier
Pleasanton, California
Gurgle@aol.com

Dear *DDJ*,

I read Alan Cooper's article "Goal-Directed Software Design" (*DDJ*, September 1996) with interest, having previously read his book *About Face*. Generally speaking, what he writes expresses clearly what I feel, although we occasionally disagree.

The disagreement at hand concerns the phrases on page 18 (*DDJ*) "...an embarrassment of compute-cycle riches." My biggest complaints about the software I use are generally: 1. bugs, and 2. slowness. The article didn't talk a lot about the Bane of Software—bugs—but maybe they're more of an implementation than design issue (maybe not always, though). But on slowness, he doesn't even seem to think it exists!

It has been my experience that no matter how fast the hardware, I end up waiting, wondering what on earth the programmers have cooked up to waste the time. I guess we agree to the extent that nowadays I tend to blame the software more than the hardware.

Stuart Ambler
Longmond, Colorado

***Java Response***

Dear *DDJ*,

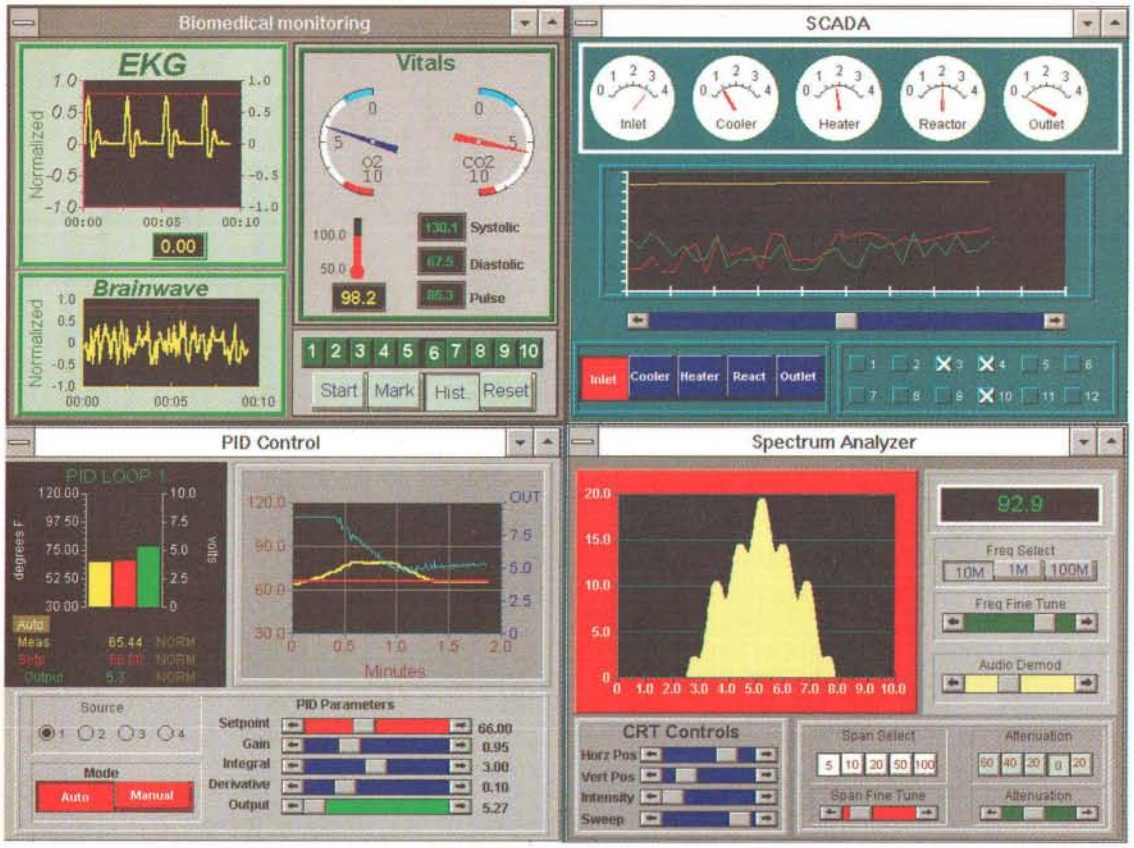
I would like to take this opportunity to respond to some criticisms of Java that were made by Luca Passani in his letter in the September 1996 *DDJ*. There are a number of criticisms of Java that Luca makes, that are inaccurate and misleading.

Luca's statement that Java applets can't and never will be able to write to the local filesystem is incorrect. The current restriction on access to the local filesystem is not a restriction imposed by Java, but is rather a restriction imposed by Netscape's Navigator. Web browsers can create security managers which allow for any degree of granularity of access to local system resources—Sun's HotJava browser, for example, allows applets to write to directories on a user-specified access list; future browsers could implement more complex security managers. For example, a security manager could be written that allows applets that are loaded from specific hosts, carry a specific public key signature, and so forth, access to specified directories in the filesystem. Luca's contention that the only alternative to "no access to the local filesystem" is "full access to the local filesystem" is incorrect.

Luca also contends that no "serious applications" have been written in Java, and that Java is too slow to download and run. His first claim is patently false; I point to Sun's HotJava browser and Java WorkShop development suite as examples of "serious applications" written in Java. The fact that there are many "eye candy" Java applets out there is a function of the fact that, until corporations start adopting Java, hobbyists (such as myself) will continue to be the primary users of Java. However, that trend is changing rapidly, and I

# Real-Time Graphics Tools *for Windows®*

*Program 16 or 32-bit applications using Rev. 2.5*



## *General Features*

- ☒ 16 and 32 bit DLLs
- ☒ Compatible with Windows 3.1, Windows 95 and Windows NT™
- ☒ Supports VisualC++™, Borland C++™, Visual Basic™, Borland Delphi™
- ☒ Royalty Free
- ☒ Bitmap and metafile support
- ☒ MDI, OWL and MFC compatible
- ☒ Print to any Windows supported printer
- ☒ DDE support

## *Dynamic Plots*

- ☒ Scrolling graphs
- ☒ Scrolling graphs with variable sample intervals
- ☒ Sweep graphs
- ☒ Logic graphs
- ☒ Scrolling bar graphs
- ☒ XY plots
- ☒ Dynamic bar graphs
- ☒ Dynamic text
- ☒ Text messages
- ☒ Alarm messages
- ☒ Annunciator panels
- ☒ Needle, arc, and pie meters
- ☒ 32K points per trace
- ☒ 32 traces in a single graph
- ☒ Historical data buffers
- ☒ Auto scroll graphs
- ☒ Formatted PID bar graphs
- ☒ Auto bar graphs
- ☒ Auto text displays
- ☒ Alarm checking and reporting
- ☒ Dynamic time axis

## *Static Plots*

- ☒ Line plots
- ☒ Horizontal and vertical bar graphs
- ☒ Scatter plots
- ☒ Pie charts
- ☒ Floating bars
- ☒ Group plots
- ☒ High-low-close plots
- ☒ Area plots
- ☒ Error bar charts
- ☒ Auto axes scaling
- ☒ Number of data points/ lineplot can exceed $10^7$
- ☒ Mix different plot types within a graph

## *Miscellaneous*

- ☒ Zooming
- ☒ Data cursors
- ☒ Scroll bars and buttons
- ☒ LED indicators
- ☒ Axes labels in numeric or time formats
- ☒ Edit charts with dialog boxes or editing functions
- ☒ Multiple x or y axes
- ☒ Multiple scalings
- ☒ Legends
- ☒ FFT
- ☒ PID control
- ☒ Mouse support
- ☒ Data compression for fast display updates

The ***Graphics Class Library*** for MFC, a class library version of the Real-Time Graphics Tools, is also available as a $200 add-on.

***Request or download our demo disk!***

***Telephone:*** 617/ 449-6155
***BBS #:*** 617/ 449-4783
***WWW:*** http://www.quinn-curtis.com
***FTP:*** ***at*** ftp.webcom.com ***in*** /pub/quinn/demofiles

| | |
|---|---|
| **Real-Time Graphics Tools *for Windows* DLL** | *$600* |
| **Real-Time Graphics Tools *for Windows* DLL & Source** | *$1200* |

***Specify C/C++, Delphi or Visual Basic version when ordering***

**QUINN-CURTIS, INC** • 35 HIGHLAND CIRCLE • NEEDHAM MA 02194 USA • TEL **617/449-6155** • FAX **617/449-6109**

*(continued from page 10)*

see this balance shifting rapidly over the next year or two.

Luca also compares the relative complexity of Java (requiring "professional developers for not so complex applications") with the simplicity of HTML. This is a flawed comparison, as HTML is not a programming language. A much better comparison would be to compare Java with C++ or Perl, both of which require developers with some degree of skill for even fairly simple programs. Users who are simply building web pages can download and use any number of prewritten Java applets (the Gamelan repository is one such source of applets). Custom solutions have always and likely will always require professional developers.

Also, Luca makes the statement that it is better to use other RAD tools and TCP/IP client/server approaches than to "waste time and money trying to make your requirements fit the Java paradigm." I utterly fail to see how the "Java paradigm," as he puts it, is any different from any other OOP programming paradigm. In fact, it is my experience that developing complex TCP/IP client/server applications in Java is significantly easier than developing them in C or C++. The language is somewhat different, but there is no clear advantage to other RAD solutions, except in the area of GUI builders. And the drag-and-drop GUI builder (such as Delphi's, for example) is already available for Java from at least one vendor. (Rogue Wave's JFactory.) I do not see a disadvantage to using Java here, and in fact, Java has several advantages in terms of security and network programming.

I will concede that there are some bugs in the current implementation of the AWT toolkit. However, as time goes by (Java is, after all, barely a year old), and these issues are resolved, Java will become that much stronger. Luca's criticism of the speed penalties for downloading and running Java applets can and will be resolved as both Internet bandwidth increase and mechanisms are developed for caching applets. Luca's criticism of "...having to download a Java .PDF-life reader every time you encounter a .PDF document" only remains valid if you assume that you have to download the applet every time. If a mechanism could be created whereby the applet could be downloaded and stored persistently on your local hard drive (and such a mechanism is not difficult; Netscape's Navigator already does some caching of applet data), this would turn into an advantage rather than a disadvantage. (I'd rather download 200K of compiled Java code than a 4-MB ZIP file, which I had to install on my system, for example.)

In summary, I feel that Luca's criticisms of Java are based on an incomplete understanding of exactly where the limitations in the current implementations of Java-based technologies lie, and that his comments are extremely misleading. The majority of the limitations currently encountered in Java applets are limitations not of the applet, nor of Java, but of either the web browser or the hardware network connection. Java certainly does have some limitations, which will be solved as more people adopt Java and more effort goes into resolving those limitations, but the arguments that Luca makes against Java are incorrect and misleading.

Matthew Cravit
Palo Alto, California
mcravit@best.com

**DDJ**

DDJ *welcomes your comments and suggestions. Send letters to* DDJ, *411 Borel Ave., San Mateo, CA 94402-3522. We can also be reached via editors@ddj.com, 76704.50@compuserve.com, and by fax atr 415-358-9749. Please state your name and address.* DDJ *reserves the right to edit letters for length and/or content.*

## There Are Enough Details in the Average Software Project To Make Your Head Spin.

**Take Control with Track Record!®** **The award-winning development tool for tracking bugs, features, releases, and more.**

If you're a software developer, project manager, or QA engineer, you need an easy, reliable way to:

- Keep track of what's been done, what's left to do, and where the problems are.
- Stay on top of bugs, features, and changes.
- Figure out who did what, when, and why.

In other words, you need Track Record 3.0, the top-rated development tool from the people who created the revolutionary BRIEF® editor.

**Track Record keeps you on top of the details.**
Track Record not only tracks bugs from first report through reproduction, fixing, and testing. It also puts you in control of the entire development cycle, helping you improve software quality. It lets you track feature requests...the release a bug fix is in... bug reports by priority, developer... and virtually anything else you're interested in. It even keeps a running history for future reference.

Most important, Track Record is easy to use. With pre-designed data types, reports, charts, and more, you're ready to take control right out of the box. Working in teams is easier, too. Why? Because Track Record keeps everyone up to date — *automatically*.

What's more, Track Record's fully customizable, object-oriented database gives you the power to track any kind of information. You can even track bugs across several projects at once.

**Order now — and find out why the experts rave about Track Record.**
Track Record helps all kinds of people take control of the details — and write better software. People like Neil Soane of Mindscape International, who says: "Track Record helps us keep our support people connected and helps us focus on improving our software. We couldn't do without it."

And the trade press definitely agrees: "Track Record is clearly the technology leader — the most flexible and sophisticated of the bug trackers." — Data-Based Advisor, June, 1995

**To order, call:**
**800-343-7308** *or* **617-267-9743**

**Just $195**
*Risk-free, money-back guarantee*
*Call for details on client/server version*

SOFTWARE Development PRODUCTIVITY AWARD 1995

TRACK RECORD

Track Record is a registered trademark of UnderWare, Inc.

UnderWare, Inc., 321 Columbus Avenue, Boston, MA 02116 USA • Fax 617-424-1839 •http://www.uw.com

CIRCLE NO. 112 ON READER SERVICE CARD

## "A RAD Tool That Aces The Competition"

*Peter Coffee, PCWeek, June 10, 1996*

"Optima++ marks a leap forward over traditional C++ tools." *Rich Dragan, Windows Sources, June 1996*

"Superb integration; innovative component-oriented tools; superior execution speed; nonproprietary language." *Peter Coffee, PCWeek, June 10, 1996*

"Powersoft's new development tool makes creating corporate applications with C++ an optimal experience." *Dan Rogers, Software Development, September 1996*

"With Powersoft's Optima++ visual tool, C++ development for Windows has never been easier." *Sean Gallagher, Information Week, May 6, 1996*

"A C++ Tool That Cures VB Envy" *Rick Grehan, BYTE, October 1996*

"Optima++ is a powerful product that combines the ease of use of VB's component metaphor with the speed and power of C++." *Steve Jackson, Visual Programmer++, September 3, 1996*

"Optima++ provides direct ODBC support, meaning classes and bound controls call the ODBC API directly, without an intermediate layer such as VB's Jet engine or Delphi's IDAPI." *Steve Jackson, Visual Programmer++, September 3, 1996*

"Powersoft has scored a home run." *Dan Rogers, Software Development, September 1996*

"Optima++ represents a breakthrough in visual programming not only for C++ developers, but for all users of fourth-generation language (4GL) client-server ment tools." *Rich Levin,*

"[Optima++] puts the fastest compiler that we've found to date into an environment that is both approachable and productive." *Peter Coffee, PCWeek, March 18, 199*

## *Experts agree. Optima++ comes out on top.*

- *Component-based RAD C++*
- *Drag-and-drop programming*
- *Over 220 components and classes*
- *Build and exploit Powersoft® DataWindows*
- *Native drivers for Sybase®, Microsoft, Oracle, Informix, DB/2, and more*
- *Scalable Sybase® SQL Anywhere™ database*
- *Exceptionally tight, fast code*
- *Build and debug CGI, NSAPI, and ISAPI custom application servers*
- *Powersoft ObjectCycle™ for team development*

PCWEEKLABS ANALYST'S CHOICE JUNE 10, 1996 Optima ++ Developer 1.0

SOFTWARE Development Incredible Product

BYTE BEST

Now you can deliver extraordinary solutions — at an extraordinary speed. Optima++™ revolutionizes development by allowing you to quickly build client/server and Internet applications using visual component assembly, drag-and-drop programming and the full power of C++.

For client/server development, the new DataWindow™ control gives you point-and-click database access, powerful extended attributes, and presentation capabilities made famous in PowerBuilder®. For Internet development, Optima++ delivers visual component assembly and seamless remote debugging so you can create high-performance applications and custom application servers.

Choose the edition that's right for you. Optima++ Developer delivers approachable C++ for client/server development. Optima++ Professional adds powerful features for the corporate developer including DataWindow technology and Internet development. Optima++ Enterprise takes scalability and performance to the next level with native database drivers and ObjectCycle™ for team project management.

So don't just read about this revolutionary RAD tool. Download your Test Drive edition today and let Optima++ take your development to new heights!

**Powersoft.**

**Get Optima++ Developer, Professional, or Enterprise today: 1-800-395-3525 or www.powersoft.com**

CIRCLE NO. 295 ON READER SERVICE CARD

the Programmer's Paradise® catalog

**Best Sellers**

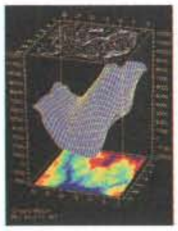
| Paradise No. | Discount Price |
|---|---|
| Dan Bricklin's demo-it! v2.0 WIN 3 1/2 LIDSD31-FT | $299 |
| Delphi v2.0—New User CD-ROM BODPT88-FT | $95 |
| Doc-To-Help v2.1 WTDH231-FT | $320 |
| HCL Exceed v5.1.3 WIN/95/NT HBHEX88-FT | $369 |
| Mathematica v3.0 WIN95/NT WOMM388-FT | $1245 |
| MS FrontPage 97 with Bonus Pack MSFPB88-FT | $139 |
| MS Mastering Internet Development CD-ROM MSIND88-FT | $95 |
| MS Office 97 Version/Competitive Upgrade MSOSD88-FT | $229 |
| MS Proxy Server MSPRX88-FT | $995 |
| MS Visual Basic Pro v4.0 WIN CD-ROM MSV4P88-FT | $469 |
| MS Visual C++ Subsc. v4.0 CD-ROM MSVC4S8-FT | $479 |
| MS Visual FoxPro v5.0 Pro Upgrade MSFPF88-FT | $265 |
| MS Visual J++ v1.0 32-bit CD-ROM MSJPP88-FT | $89 |
| MS Visual SourceSafe v4.0 MAC/WIN MSVSS88-FT | $479 |
| MS Windows NT v4.0 Workstation CD-ROM MSWIA88-FT | $299 |
| MS Windows NT v4.0 Workstation V+C Upgrade MSWIB88-FT | $139 |
| MS Visual C++ v4.2 Enterprise Upgrade MSVCF88-FT | $469 |
| MSDN Professional Subscription WIN 95 CD-ROM MSDNC88-FT | $495 |
| Multi-Edit v7.1 Windows AMCME31-FT | $139 |
| PowerBuilder Desktop 5.0 WIN CD-ROM PWSPG88-FT | $189 |
| PVCS v5.2 WIN/NT/95 3 1/2 ISPVP31-FT | $449 |
| SuperCede v1.0 Java Edition ASSUP88-FT | $99 |
| System Commander v3.0 3 1/2 VCSC331-FT | $69 |
| VB Assist v4.0 SDVBD88-FT | $125 |

Call for shipping charges/return policy. Prices subject to change without notice.

## *Celebrate President's Day With These Top-Sellers!*

### Olectra Chart
*by KL Group*

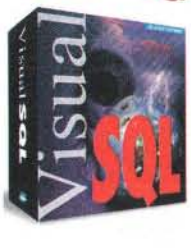
Setting new standards for functionality and performance, Olectra Chart lets developers add stunning 2D and 3D graphs to their Windows applications. Olectra Chart offers developers more control over graph colors, styles and properties along with features like dynamic zooming, scaling and rotation. Available as a 16- and 32-bit OCX and includes a fully documented C/C++ API and MFC and OWL wrappers.



**Paradise No.**
**KLOLF88-FT**
**$239**

### Visual SQL®
*by Blue Sky® Software*

New! Visual SQL turns Microsoft Visual C++ and Visual C++ Enterprise Edition into full-blown Client/Server development environments. Visual SQL integrates seamlessly with Microsoft Developer Studio using intuitive wizards to generate MFC Client/Server code. Visual SQL builds on the power of Visual C++ and offers developers the best of all worlds: the rapid application development of visual tools such as Visual Basic and the high-performance of compiled C++ code.



**Paradise No.**
**BLVSQ88-FT**
**$1395**

### DbCAD dev 1.42 16/32-bit DLL & OCX Royalty-free
*by ABACO srl*

Ingenious, powerful and easy-to-use library for Windows 3.x, 95 and NT for vector database management. Manages graphic window with zoom, pan, pick and overview commands, compatible with most standard vector and raster formats including DWG, DXF, WMF, TIF, BMP, RLE, RLC; also permits their display using transparency effect; partial loading of drawings. All that with intelligent functions to link your databases with each vector entity, allowing you to extend your application to support CAD, GIS, DTM, TDM and FM functionality. Royalty-free.

*New Version 1.42*

**Paradise No.**
**ABDBE31-FT**
**$1399**

### VTune
*by Intel Corporation*

Intel's Visual Tuning Environment for Windows®

- Visual & Easy To Use Performance Methodology
- System-Wide Monitoring
- Identifies Performance Hotspots
- Detailed View of Software Execution
- Optimization Advice

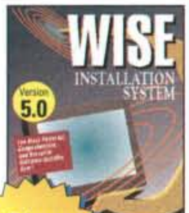
**Come see us at SD '97 West—Booth #2126.**

*Supports MMX™ Technology*

**Paradise No.**
**INTVT88-FT**
**$129**

### WISE Installation
*by Great Lakes Business Solutions*

Creates professional installations for Windows, Windows NT, and Windows 95 all with a single installation script. The WISE Installation System includes both 16- and 32-bit versions of the integrated development environment. WISE includes all of the standard installation features plus: custom dialog/graphic editors; CD-ROM/Network installations; multi-lingual support; and full Windows 95 install requirements.



*New Version*

**Paradise No.**
**GLISV31-FT**
**$169**

### WinSCSI-32
*by Sequoia Advanced Technologies, Inc.*

Full 32-bit SCSI command link library with over 150 functions designed to make your SCSI application programming tasks easier. Complete with extensive sample source code and electronic online documentation, WinSCSI-32 is the ideal choice for all your SCSI programming needs. For Windows 95 and Windows NT. Royalty-free.

**Paradise No.**
**SEQWS88-FT**
**$175**

### c-tree Plus®
*by FairCom*

DOS • WINDOWS • NT • UNIX • OS/2 • SUN • RS6000 • HP9000 • MAC • QNX • BANYAN • SCO.

This well known, highly portable data management package has become established as the tool of choice for commercial development. Offering unprecedented data control, choose from direct low level access, ISAM level, or ODBC access with the FairCom Server. Single User, Multi User, or optional Client/Server, ANSI Standard. Full Source. No Royalties.

*Full Win 95 support!*

**Paradise No.**
**FACTP31-FT**
**$785**

### Spread v2.5
*by FarPoint Technologies*

A complete spreadsheet control for most environments that support a VBX, OLE control, or DLL. Use as a spreadsheet to obtain variable lines of data or display tables of information. Data-aware—connect to databases with the Access Engine and ODBC. Includes over 250 properties and our improved Spread Designer. Formulas, sorting, and full-print support too.

**Paradise No.**
**FASVX31-FT**
**$229**

***Order online 24 hours a day, every day at www.pparadise.com***

## Available at the best prices from
# *Programmer's Paradise®*
***Co-sponsors of Microsoft Developer Days***

### WinHelp Office™
*by Blue Sky® Software*

WinHelp Office includes everything you need to quickly and easily create professional Help systems for Windows 3.1, Windows 95, Windows NT 3.51, and Windows NT 4.0; the new HTML-based Help; printed documentation; and intranet/Internet Websites—all in one box! WinHelp Office is power-packed with the award-winning RoboHELP, Mastering WinHelp, WinHelp Video Kit, WinHelp Tool Kit, WinHelp HyperViewer, and Moving to HTML Kit.

**Paradise No. BSWOF88-FT**

**$679**

### RoboHELP®
*by Blue Sky® Software*

The award-winning RoboHELP 4 is the fastest and easiest way to create Help systems for Windows 3.1, Windows 95, Windows NT 3.51, and Windows NT 4.0.; plus the new HTML-based Help; printed documentation; and intranet/Internet Websites. Full support of ALL Windows Help features, plus RoboHELP gives you the power and ease of use of Microsoft Word. RoboHELP saves you time—it's fast. Best-seller!

**Paradise No. BSRHF88-FT**

**$479**

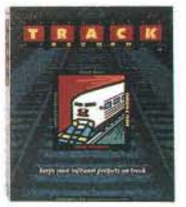
### Visual Developers Suite Deal
*by Visual Components*

The Visual Developers Suite Deal is better than ever. Now with four full OCX tools for developing great apps. Includes the latest versions of Formula One for dynamic spreadsheets, First Impression for stunning charts, VisualWriter for robust text editing, and VisualSpeller for powerful spell checking. It's our best deal yet.

**Paradise No. VTVSD31-FT**

**$269**

### Track Record® for Windows
*by UnderWare, Inc.*



Track Record is the top-rated tool that tracks every aspect of your software development project: bug reports; feature requests; releases; testers; and much more. As *Data Based Advisor* noted, "Track Record is clearly the technology leader—the most flexible and sophisticated of the bug trackers." Designed for individuals and for workgroups sharing information on a network. New version 3.0 offers integration with version control software, optional e-mail notification, summary reports and more. Check out this winner of Software Development Magazine's Productivity Award!

**Paradise No. UWRTB31-FT**

**$169**

### Codewright 4.0
*by Premia*

More powerful than ever with the new API Assistant! Prevents bugs before they happen; has Button Links to non-text files; and Direct Access version control interface. Multi-file search and replace, help topic scanner, file differencing and merging, are real time-savers.

**Paradise No. PCCW488-FT**

**$199**

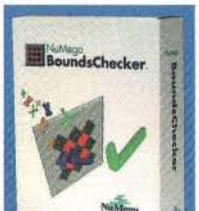
### Tab Pro v2.0
*by FarPoint Technologies*

Add organized and professional user interfaces to your Windows applications without spending a lot of time and money. Tab Pro allows the quickest and most flexible presentation using any of the supplied styles—no matter what your design environment. Customize tabs and notebook pages using Tab Pro's 150 properties, 15 events, and 9 functions.

**Paradise No. FTTBP31-FT**

**$89**

### BoundsChecker™
*by NuMega Technologies*

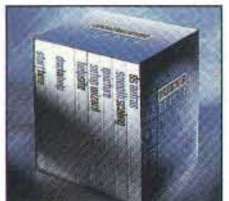


BoundsChecker is the award-winning Advanced Error Detection™ system for Windows C/C++ and Delphi developers. BoundsChecker is built specifically for Windows architectures to constantly find thousands of bugs that ordinary memory-checking tools cannot. There's NO re-compiling, NO re-linking, and NO extra steps, and BoundsChecker supports the latest Windows technologies like ActiveX, COM/OLE, ODBC, Internet APIs and more.

**Paradise No. NUMCB31-FT**

**$499**

### Documentation Studio
*by WexTech Systems, Inc.*



**NEW!**

Documentation Studio is an integrated suite of authoring tools and utilities that allows simultaneous creation of printed documents, Windows online Help, standard HTML pages and Microsoft HTML Help, all from a single source file. Documentation Studio includes Doc-To-Help 2.1, Helpsite 1.5, Quicture 2.0, Smooth Scaling 2.0, Setup Wizard 2.0, Lotus ScreenCam and BookMaker ClickBook.

**Paradise No. WTDOC88-FT**

**$785**

**On March 19th**

***Microsoft®* presents**

All the tools • All on one day

Microsoft® **Developer** Days

All around the world!

**$89** if you register by 2/13/97

**For more information visit www.pparadise.com**

Let the products featured on these pages—or any of the other outstanding products featured in the Programmer's Paradise catalog—lead you to "greatness" in the software development world. And, take advantage of our outstanding service—personally guaranteed by our president, Roger Paradis!

### *A Presidential Guarantee...*

To place your order, call today and take advantage of our convenient, quick 2-day delivery to any location in the continental U.S.!

"If you're not happy, we're not happy! You have my personal guarantee that we'll provide you with outstanding products, support and service."

**–Roger Paradis**
President & CEO
Programmer's Paradise, Inc.

**ORDER NOW**
***or call for your free catalog!***

**1-800-445-7899**

**See us at SD '97 West Booth #1002**

CIRCLE NO. 128 ON READER SERVICE CARD

*To Order Call 800-445-7899 or FAX 908-389-9227*

# Dr. Dobb's News & Views

## Electronic Commerce

Purdue's Markus Kuhn kicked off the Second USENIX Workshop on Electronic Commerce with a talk on smart card tamper-resistance (or lack thereof). Kuhn (whose paper, cowritten by Ross Anderson, was selected as the conference's best paper) refuted any assumption of a smart card's security by demonstrating practical techniques for tampering with common smart cards. Other papers at the conference included Openmarket's Daniel Geer on token-mediated certification, Stanford's Steven Ketchpel on a universal payment API, Carnegie-Mellon's Darrell Kindred on automatic protocol checking, and Bell Labs' Eran Gabber on a distributed protocol for electronic commerce.

There were some very good higher-level talks as well. Attorney Benjamin Wright spoke on the legality of digital signatures, comparing and analyzing existing law. John du Pré Gauntt, the business editor of *Public Network Europe* and a researcher at the London School of Economics, discussed electronic commerce in a social context, describing how electronic commerce changed the very way we think about money, offering several interesting historical examples. UC Berkeley Law Professor Pamela Samuelson, well-known for her work in intellectual-property law and as a legal columnist for *Communications of the ACM*, discussed intellectual-property law's effect on commerce law, citing current legal cases as examples of the direction and implications of current public policy.

*—Eugene Eric Kim*

## PC to Mac

Executor 2 from ARDI lets you run many Macintosh applications on a PC. The system, which runs on Windows 3.1/95, OS/2 Warp, DOS, and Linux, was developed using "clean-room" techniques, and includes no Apple code. It can also read/write Macintosh-format floppy disks, CD-ROMs, and hard drives. A free demonstration version is available on ARDI's web site (http://www.ardi.com/).

*—Tim Kientzle*

## Computers and the Law

The ever-changing issues of server security, cryptography, trade secret prosecution, and first amendment interpretation were hot topics of discussion at the Sun User Group's Computers and the Law III symposium.

Among the interesting computer law cases reviewed was a puzzling new precedent set regarding real-time information and the commercial appropriation of "fresh" news. Jim Hemphill of George, Donaldson & Ford L.L.P. discussed *The National Basketball Association (NBA) v. Sports Team Analysis and Tracking Systems Inc. (STATS)* case, in which the NBA sued Motorola and STATS over the practice of distributing real-time basketball scores to subscribers via pagers. The federal courts ruled in favor of the NBA, determining that, though NBA scores and statistics could not be copyrighted, the NBA does "own" real-time information about its games. Defining freshness as a commercial property could introduce some interesting new intellectual-property concerns.

Another discussion of note was Silicon Valley criminal defense attorney Tom Nolan's discourse on a recent amendment to California's Uniform Trade Secrets Act. The amendment broadens the state's definition of a trade secret, potentially lifting some trade secret disputes from the civil arena and enabling district attorneys to prosecute them under criminal law.

*—Deirdre Blake*

## Pass the Quarters

You may think that managing a team of programmers is hard work, but to some people it's just a game. MCI Systemhouse and Thinking Tools have developed an "agent-based simulation tool" (read "video game") in which you play a bedeviled manager who must deal with cranky programmers, uncooperative clients, and unsupportive upper-level management. With a $1495 price tag, it's unlikely to outsell Sonic the Hedgehog, but the developers hope large companies will see it as a training investment.

*—Tim Kientzle*

## E-mail Directory

Later this year, both Pacific Bell and Nynex will begin allowing phone customers to include e-mail addresses and URLs with their white-pages phone-number listings. The service won't be free for Pac Bell customers. Prices are expected to fall in line with the fees current customers pay for adding, say, fax numbers to their listing—$5.00 setup and 85 cents a month.

*—Amy Wu*

## MSN2 Alienates MSN Members

Microsoft's new, more public, online service, MSN2, has drawn criticism from disillusioned users of the old Microsoft Network service. Grievances are being posted at a protest site called "The Official MSNot Hate Site" (http://www.geocities.com/SoHo/9120/). The protesters' gripes range from alleged censorship (critical comments have reportedly been removed from MSN2 bulletin boards by forum hosts at Microsoft) as well as MSN2's heavy use of ActiveX graphics, which makes browsing a slow process for some users.

*—Deirdre Blake*

## Smart Money

When it comes to chips, the 125,000 attendees at last fall's Comdex in Las Vegas were apparently thinking "Pentium" or "PowerPC" instead of "roulette" or "blackjack." To the disgust of some casino operators, gambling was down during the first two days of the show.

*—Jonathan Erickson*

## The Season of Sharing

Microsoft topped all other U.S. companies in gift giving, donating $73.2 million in fiscal year 1995. The bulk of the company's largess—$62.1 million—was in software donations. Considering the tax write-offs and chance to hook new customers with upgrades, you have to wonder where the philanthropy part comes in.

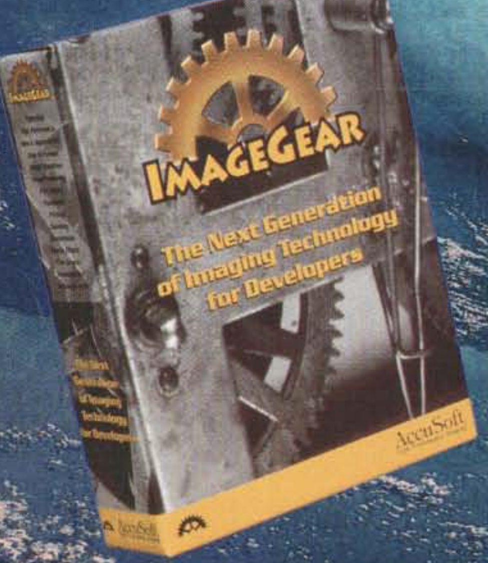
*—Monica Berg*

## Virtual Prayers

Virtual Jerusalem (http://virtual.co.il/) has launched the Send-a-Prayer service by which visitors to the site can e-mail prayers to the Holy Land. The staff at Virtual Jerusalem will print out prayers on a daily basis and place them in cracks in the Kotel—the Western Wall. Placing prayers at the holy site is an age-old Jewish tradition and, according to Virtual Jerusalem, the practice "lends permanence to your prayer." The site also features news about the city, a calendar of events, and interviews with famous international Jewish personalities.

*—Deirdre Blake*

# The Most Comprehensive Imaging Toolkit on the Planet™



- Over 45 File Formats
- Cross Platform
- High Performance
- Scan, Display & Print
- Document Imaging
- Scale-to-Gray
- Image Processing
- Special Effects
- and Much More!

## The Next Generation of Imaging Technology for Developers™

ImageGear™ 6.0 shifts your development schedule into warp drive by providing you with the most comprehensive, easiest to use image processing solution available. With support for over 45 file formats and an API featuring 200 plus functions, ImageGear is the toolkit that no developer should be without.

Standard and Pro Gold™ versions allow you to purchase the level of performance and functionality you need. ImageGear offers the high speed display of both bitonal and full color images with functions like scale-to-gray, automatic aspect-ratio correction, panning window, magnify window, and much more. Advanced features like auto-deskew, sub-pixel display accuracy, rotate to any angle, and special effects add robust image processing to any application. ImageGear's new display technology is fast and easy to use and printing is more accurate than ever with new high-level functions. The TWAIN scanning interface with support for 32-bit drivers gives you complete control. GUI functions make integration quick and seamless, and multi-platform support provides a complete solution to portability issues.

The AccuSoft Image Guarantee™ assures that your application will read every valid image. You can turn to us with confidence because we have over ten years of experience as the leader in quality image processing. So call the experts at AccuSoft, and step up to the next level in quality and performance with ImageGear.

**Visit our Website @ www.accusoft.com**

- FREE ImageGear 6.0 Demo
- Comprehensive Product Info
- ImageGear Special FX in Action
- Imaging Toolkit Glossary and Much More!

**AccuSoft Corporation**
Two Westborough Business Park
Westborough, MA 01581
Tel (508) 898-2770
FAX (508) 898-9662

AccuSoft®
*High Performance Imaging™*

30™ MINUTE Delivery

***Call for FREE Demo! Toll Free: (800) 741-7494***

DD5

**CIRCLE NO. 65 ON READER SERVICE CARD**

**Versions Available**

Windows
Win95/NT
MIPS NT
Alpha NT
VBX
ActiveX(OCX)
OS/2
SUN OS
Solaris
HP-UX
AIX
SGI
SCO
MAC
PowerMac

**Over 45 File Formats**

TIFF
JPEG
PCX
JFIF
TGA
Group 3
Group 4
PGM
DIB
IMT
DCX
BMP
KFX
RLE
LV
CALS
ATT
CLP
XWD
XBM
IFF
SUN
PNM
IOCA
GX2
XPM
ASCII
CUT
GEM
BRK
MAC
PSD
MSP
PNG
PCD
IMG
IGF
SGI
ICO
PBM
PPM
MO:DCA
WMF
WPG[1]
PICT[1]
EPS[1]
GIF[2]
ABIC[3]
CIF[3]
JBIG[3]
DICOM[3]
and others

ACCUSOFT IMAGE GUARANTEE™

# Unit and Regression Testing

*Reducing bug counts isn't as hard as you think*

Adrian McCarthy

While we were looking at an image generated by my ray tracer, my brother asked, "How do you know it's right?" The question sparked an interesting discussion on software testing. Until then, I had done only black-box testing: Create a scene and inspect the output for inconsistencies. One bug in my ray tracer—a sign error in a vector cross-product function—eluded detection for nearly a year because the symmetries in my early test scenes caused the sign error to cancel itself. Even after detection, it took a long time to find the bug. It could have been detected easily (and the bug found and fixed more quickly) by testing the vector operations in isolation from the rest of the system.

Of course, black-box system testing is an important step in making sure your program works, but it's only one piece of the puzzle. Unit testing (testing a function, module, or object in isolation from the rest of the program) and regression testing (rerunning tests to detect unexpected changes in behavior) can dramatically reduce your bug counts.

In his excellent book, *Writing Solid Code* (Microsoft Press, 1993), Steve Maguire mentions unit testing, but he doesn't give many details on how to set up such tests and use them. In this article, I describe how to build simple and effective unit tests and how to automate regression testing using make.

Unit testing should not be a luxury accessible only to huge product teams with hordes of testers. In fact, unit and regression testing make it possible for individuals and small teams to outperform big companies that rely too heavily on black-box testing to ferret out bugs.

## Automating Unit Tests

Suppose for each key object or module in your project, you wrote a small test program to test just that portion. That might sound like a lot of work, but, as you'll see, it's not. If each of these unit tests accepts sample data from standard input and sends the results to standard output, you can automate the tests using your favorite variety of make.

Imagine implementing a C++ class for performing vector arithmetic in a file called VECTOR.CPP. A unit test TVECTOR.CPP file would have targets in the makefile, indicating its dependency on the module it tests; see Example 1(a). The unit test is driven by a data file called TVECTOR.IN and produces output that you capture in TVECTOR.OUT, as in Example 1(b). To pull it all together, a pseudotarget generates output from all of your unit tests; see Example 1(c).

As you add more units to your project, you build more unit-test programs and add the desired output file to the TESTOUTS list. You now have a way to automatically test all parts of your program that have changed.

Big deal, you think—those output files still have to be checked by hand to see if they're right. True, but only the first time. After that, you care only about the changes in behavior. Changes occur when you make a fix, when you make a global change, or when you enhance a unit test. In all of these cases, you need only verify that the changes are appropriate. Example 2 adds an accept target, enhancing the regress target.

If your unit tests have passed, you run make with the accept target. That makes reference copies (often called "canon files") of the test results. When you make changes in the future, the regress target will compare the new test outputs with the canon files and list the differences in REGRESS.RPT. The regression report tells you immediately what changed. If the changes are correct, you simply accept them to make new reference copies. If not, you correct the problem and run the regression tests again. These makefile examples have been simplified for discussion. Listing One (listings begin on page 82) presents a more realistic example.

Have you ever considered making a global change in a big project and felt some trepidation? Perhaps you've been building and testing debug versions, loaded with assertion macros. The ship date is approaching, and it's time to disable the debug code. What if somebody accidentally put code with important side effects into an assertion? Maybe your team has been slaving away, porting code to a new compiler, memory model, or operating system. At times like these, wouldn't it be nice to type MAKE REGRESS and know in a few minutes which modules may have changed their behavior?

## Building Unit Tests

Is it worth making a test program for each unit? Yes. The effort is small, since the test program can be simple. More important-

*Adrian is a software engineer for Intuit and can be contacted at adrian_mccarthy@intuit.com.*

ly, the process will make your code better and smooth your development by eliminating bugs earlier.

What should a unit test do? That depends on the module being tested. Minimally, the test should thoroughly exercise the module, striving for complete coverage. For example, to test the *Vector* class, an exerciser would call each of the class's member functions with a list of vectors read from TVECTOR.IN (see Listing Two). Fortunately, this simplistic approach is appropriate for many modules.

Slightly more ambitious is a unit test that attempts to verify the requirements directly. For example, if you're testing a preprocessor that should strip comments and trailing spaces, then the checker would examine the output for comments and spaces that were missed.

A sophisticated unit test might try to simulate a client. A simulator makes sense for units that maintain state information from call to call, as is typical with parsers and event-driven systems. In these types of modules, the order of the calls is significant. You can have the simulator generate random calls, but that is incompatible with our regression testing, since we require the output of a successful test to match the reference run. One approach is to have the simulator use the same input-driven technique of the exerciser and have a separate program create the random data. A long, randomly generated sequence provides a good chance of uncovering sequence-dependent bugs while maintaining reproducibility.

For particularly troublesome modules that maintain state information and require intensive abuse to beat out the bugs, a combination simulator/checker might be the ticket. The simulator portion would simulate calls in a random order. Rather than outputting the results directly (and thereby making it impossible to compare to the canon files), the checker portion would verify the results and output a summary. Summaries of successful runs would always match the canon file, but one from a failure would not. One disadvantage of the simulator/checker is the amount of effort required to build it. In addition to writing code to simulate and check the module, you should add a facility to reproduce a run so that when a bug is uncovered, it can be traced. Often this can be done by supplying the seed for your random-number generator.

Once you've decided which approach to take, you need to decide exactly what your test program is going to do. Make it data driven so you can easily add more tests. Don't be afraid to let it produce reams and reams of output. Remember, you're only going to check it manually once.

To decide which calls to make and what data to use, consider the following three things:

- Your requirements list.
- The module's public interface.
- Your knowledge of boundary cases.

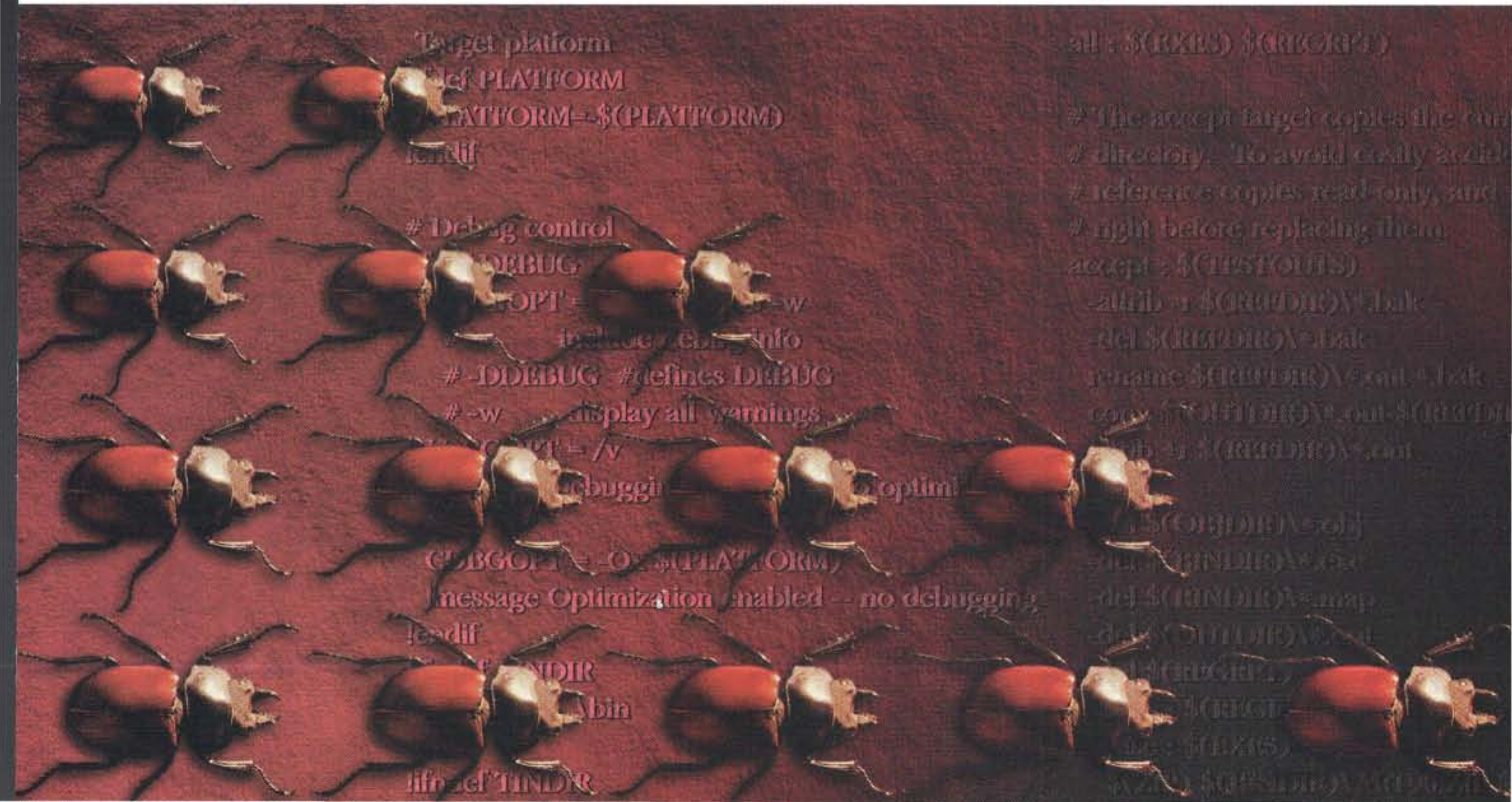
If your requirements specifically mention certain cases (like what to do if a buffer is too small), make sure your test module tries those special cases. If you don't have a formal requirements list, you still have a good idea of what the module is supposed to do. For a moment, try to forget what you know about the implementation and pretend you're a client who only has access to the comments in the header file or other documentation. The goal is to ensure that the module works as advertised.

Make sure you call every function in the public interface. For example, if you're testing a complex number object, make sure you test all of the operators like + and +=. Don't assume that one works because the other does, even if you know you implemented one by calling the other. When you add a new function to an existing module, make sure you expand the unit test to exercise it. The goal is to ensure coverage.

Finally, think about how you implemented the module. What assumptions did you make? What are the buffer limits? Where are the boundary cases? Test them by adding more test data to the input file. For my vector class, I made sure I had the zero vector, orthogonal unit vectors, vectors with lots of decimal places, normalized vectors, unnormalized vectors, and so on; see Listing Three. Again, the goal is to ensure coverage.

How many cases should you handle? How do you know when your test is complete? You'll never know for sure. When I write a new module, I work on the unit test until I find at least a few bugs. If your test passes the very first time, then you probably missed something. Stress it harder until you uncover a few bugs.

Listing Two presents the unit test for a vector class, and Listing Three shows the input file that drives the test. The program itself is trivial, but by adding cases to the input data, I eventually shook out about a dozen bugs.



The test for the vector class tries each sample vector with each of the unary operators (unitize, scale, negate, and so on), and it tests each binary operator with every pair of sample vectors. That creates a lot more tests, but that's the whole idea.

## Benefits of Regressive Unit Testing

From the start, I was convinced that unit and regression testing were worthwhile, but I was surprised at how useful they have proven to be. These are not theoretical benefits, but real advantages I observed almost immediately after implementing my first few unit tests. Unit and regression testing will:

- Find bugs early.
- Isolate bugs faster.
- Stop bugs from coming back.
- Prevent global changes from creating surprises.
- Validate interfaces.
- Increase coverage.
- Test error handling.
- Debug smaller chunks.
- Give you a deeper understanding of your code.

If you're still doubtful, consider writing just one unit test, perhaps with a module you already use or with the next one you write. While it's useful to have unit tests for all the modules in your program, having one or two is better than none, especially for the key modules. If it doesn't seem to help much and you abandon the practice, at least leave the one you built in the makefile. One day it'll catch something nasty and unexpected.

## Practical Suggestions

Some people resist the idea of unit tests because it seems that only toy modules can be tested in isolation. Usually, this just reflects a lack of creativity on the programmer's part. If you have a nontrivial module, consider it a challenge to write its unit test. Unit tests can be written to simulate asynchronous I/O events, user events, client requests, server responses, plain function calls, and just about anything else required to test a module. After all, if the rest of your program can interface to a module, then you should be able to write a different program that also connects.

Some modules may seem too dependent on other modules to be tested in isolation. Often, this occurs when there's a pipeline of modules operating on data. It might be awkward to test a parser without also calling the lexical analyzer, which calls the preprocessor. There are a couple of approaches you can try.

One is to let each stage of the pipeline rely on the earlier stages, but to write a test for each stage. For example, you could have one unit test for the preprocessor, one for the lexical analyzer and preprocessor in combination, and a third for the entire pipeline. By using the same input and checking the outputs at each stage, you can isolate changes and bugs about as easily as if you had truly independent tests.

Another solution is to have the test module provide stub interfaces for the other dependencies. For example, the test program for the parser might include an interface equivalent to the lexical analyzer. The stub would simply provide a canned

```
(a)  TVECTOR.OBJ : TVECTOR.CPP VECTOR.H
     TVECTOR.EXE : TVECTOR.OBJ VECTOR.OBJ

(b)  TVECTOR.OUT : TVECTOR.EXE TVECTOR.IN
            TVECTOR.EXE < TVECTOR.IN > TVECTOR.OUT

(c)  TESTOUTS = TVECTOR.OUT
     regress : $(TESTOUTS)
```

***Example 1:*** *(a) A unit test called* TVECTOR.CPP; *(b) the unit test is driven by a data file; (c) output from unit tests.*

```
accept:
   del *.ref
   copy *.out *.ref
regress:  $(TESTOUTS)
    echo "Regression Report" > REGRESS.RPT
    fc TVECTOR.REF TVECTOR.OUT >> REGRESS.RPT
    # ... one for each unit test
```

***Example 2:*** *Adding an accept target and enhancing the regress target.*

stream of tokens from the input file. Stubs also work with nonpipelined dependencies.

If a module has so many dependencies that the unit test is difficult, then it probably needs a unit test more than any other module. When there's that much complexity, it will have more bugs, and it'll be much easier to find and repair them if you've got some scaffolding to help you get to the heart of the matter.

When writing test programs, give a little thought to the output that will be captured and compared. Keep in mind that you'll be looking primarily at differences as reported by your favorite file-comparison program. Put enough information on a single output line to identify which case failed. Each output line from the *Vector* exerciser shows the vector or vectors used, the operator, and the resulting vector. In more abstract cases, this explicit detail might not be possible. An alternative is to refer to the line number in the input file that generated the current case.

If you get serious about unit and regression testing, take the time to integrate them into your development environment. Your test programs, sample data, and canon files should be kept in your source-control system. The ability to recreate tests from an earlier version can help isolate changes that led to elusive bugs. Establish a naming convention for your tests. In my examples, I've simply prepended the module name with T (for Test). You might want to do something more sophisticated. If you have automated weekly or nightly builds of your system, expand them to run the unit tests as well. A fancy system might even send e-mail to team leads if the regression tests indicate changes in behavior.

In my makefile examples, the accept target cavalierly replaces the current canon files with the current test results. This may be too accident prone for a large project. There are several alternative ways to handle it.

- Write a batch file that asks for confirmation.
- Have the accept target make sure the current references are recoverable by checking them into your source-control system.
- Create makefile rules to check in test results individually.

The best solution might be a combination of these approaches.

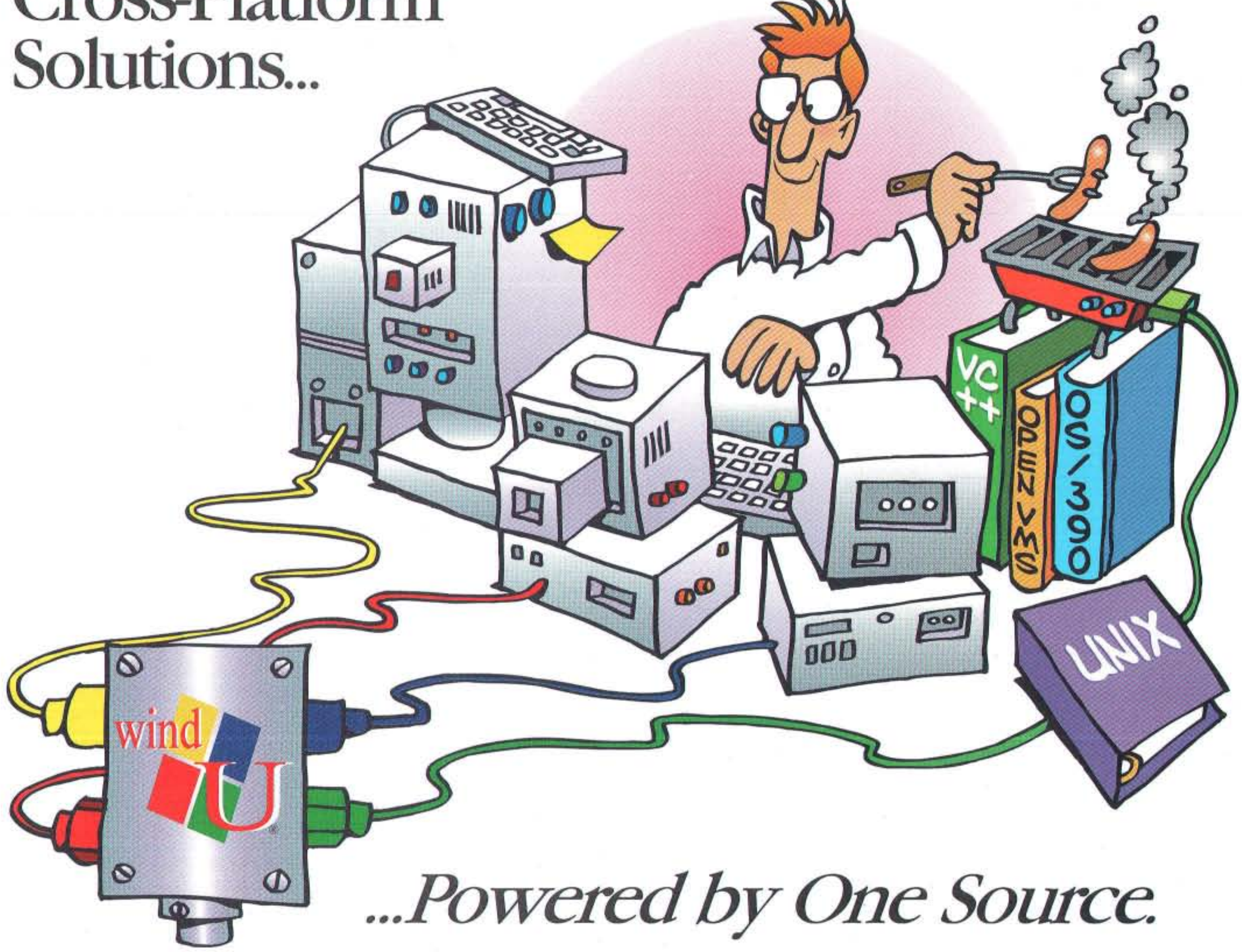
## Conclusion

Fortunately, few programmers write an entire application before testing it at all. Common practice is to write a bare skeleton and then fill in the pieces, testing along the way. But the first working skeleton may require bringing together quite a few untested pieces before having anything that runs. The program-in-progress generally isn't the best test bed for new components, nor does it provide for automation of ongoing testing. Nonetheless, common practice survives because the perceived value of better strategies doesn't justify perceived cost. My experiences, however, indicate that unit testing has more benefits than one might think, and automated regression testing of these units is easy to implement with existing tools.

**DDJ**

**(Listings begin on page 82.)**

# Cross-Platform Solutions...



## ...Powered by One Source.

Get a power-surge from your Windows code! With Wind/U® from Bristol Technology®, you can re-use your Microsoft Windows source code to deliver applications across multiple platforms in the enterprise.

Interested in increasing productivity? Spend more time creating or enhancing your core software products. Wind/U lets you produce a functionally identical UNIX, OpenVMS, or OS/390 native version in a fraction of the time and cost.

Do you need ActiveX Controls on UNIX? The new Wind/U ActiveX SDK allows you to re-use your ActiveX source code with Internet Explorer for UNIX.

To ensure compatibility with the most recent Windows releases, Bristol licenses Microsoft Windows source code and incorporates this technology into Wind/U.

Leverage the power of Wind/U, the Windows API, and MFC today. You'll get increased productivity. You'll get superior functionality. And you'll get a cross-platform application powered from a single source.

***Wind/U offers:***

- Win32 API & MFC on UNIX, OpenVMS, and OS/390
- The latest Visual C++ features
- Complete OLE and ActiveX
- Threads and WinInet APIs
- Native look-and-feel & performance
- Windows Common Controls & Dialogs
- Industry standard online help (WinHelp) & PostScript/PCL printing

***Get more information on Wind/U:***

www.bristol.com or info@bristol.com

In the U.S. call:
(203) 438-6969

In Europe call:
+31 (0)33 450 50 50

Windows NT MAGAZINE
UNIX/Windows NT Technology Award

1996 Outstanding Product UNIX REVIEW

BRISTOL
T·E·C·H·N·O·L·O·G·Y

*Delivering Cross-Platform Solutions.*

CIRCLE NO. 68 ON READER SERVICE CARD

# A Debug/Trace Tool

*Powerful debugging with minimal tools*

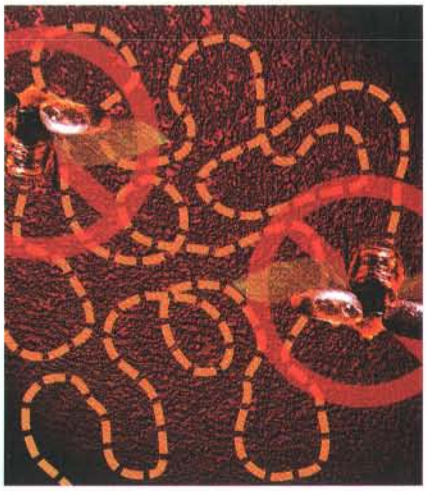
Rainer Storn

When it comes to debugging, programmers usually fall into one of two camps: those who regard *printf()* as old-fashioned and believe that modern source-level debuggers do everything necessary, and those who claim that source-level debuggers often don't. For instance, the former argue that nothing can replace features such as stepping forward and backward and the option of looking at arbitrary variables at any time. The latter, however, claim that source-level debuggers often don't let you set the granularity or skip breakpoints for a certain amount of times, that few offer CRC capabilities, that regression tests require code instrumentation anyway, and that they are tired of setting each breakpoint anew once they've changed and recompiled their sources.

The debug/trace tool I present in this article provides help for the second camp, and maybe even for the first one—if you are willing to exploit the advantages of both strategies. The tool is also appropriate when you don't have a powerful debugger available, or when you need trace functionality built into your code to support error-spotting at the client site.

*Rainer, who studied at the University of Stuttgart, Germany, is a postdoctoral fellow at the International Computer Science Institute and can be contacted at storn@icsi.berkeley.edu or rainer.storn@zfe.siemens.de.*

This tool has its roots in the debugger developed by Robert Ward in *Debugging C* (Que, 1986). However, I have changed and extended it significantly. Although my tool uses Borland C++ 3.1 and runs on a 486 PC, its parts are independent of the 80x86 family's memory organization and have run on a SPARC 5 under SunOS 4.1.3. The complete source code, sample data, and executables are provided electronically; see "Availability," page 3.



## The Debug/Trace Tool

The *dbg()* function is central to the functionality of the debug/trace tool I present here. For practical purposes, *dbg()* constitutes an extended *printf()*; it is declared as in Example 1(a) and can be used as in Example 1(b). The first variable in *dbg()*, arbitrarily chosen to be *7_stp*, is a label and will be used by the tool like a breakpoint address or a trace identifier. The second *dbg()* variable, arbitrarily chosen to be *2*, is a trace level and will be used to enable or disable the printing part of a *dbg()* statement, depending on the trace level activated by the tool. If you choose the trace level to be *2*, then all *dbg()* statements that exhibit the levels 0, 1, or 2 will perform the appropriate printout, which is defined by the subsequent parameters. These parameters follow the syntax used in a regular *printf()* statement.

## The Command Interpreter

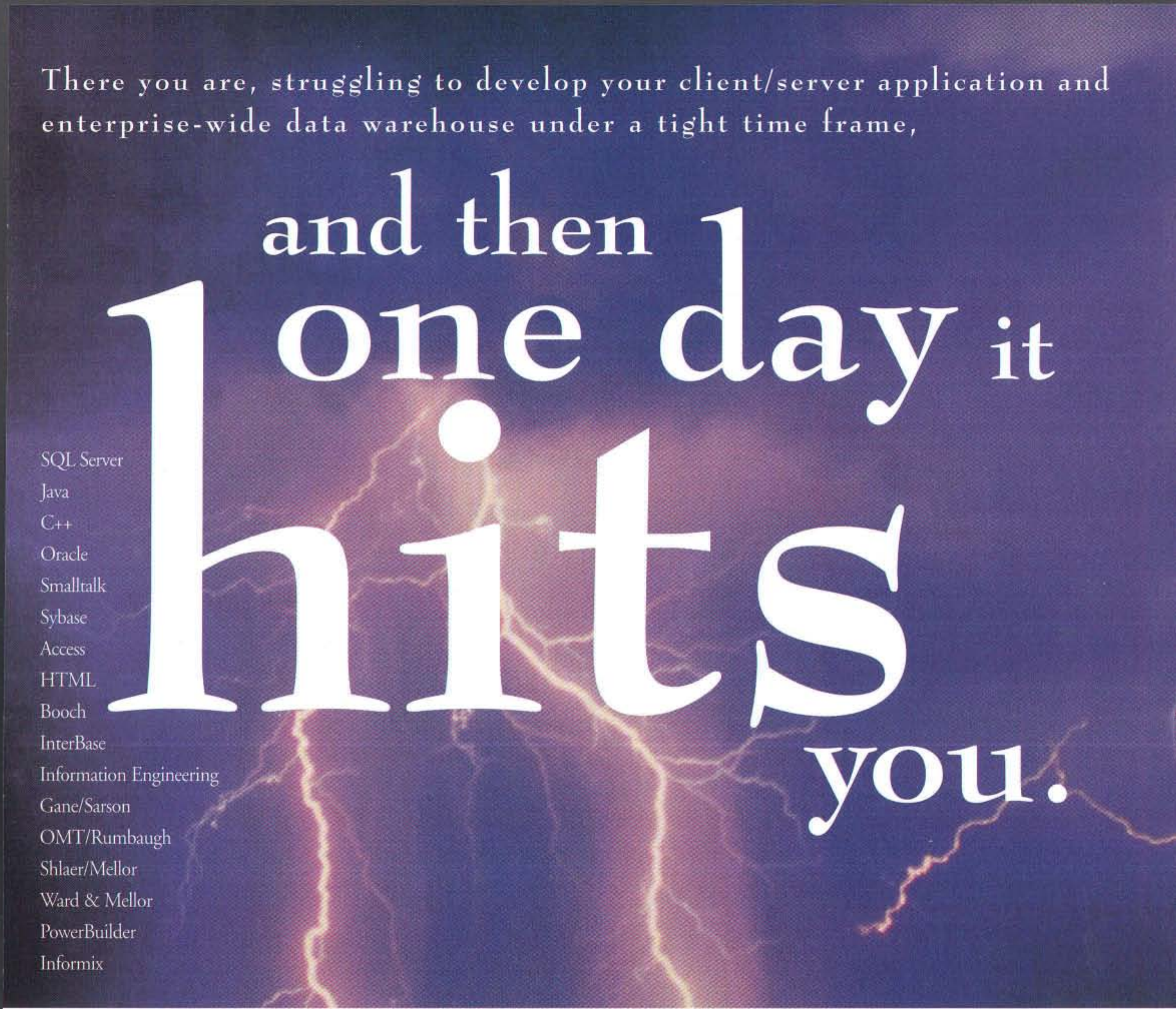
Once you've started to run your program, the tool will start to display *DBG>>* in order to give you access to the tool's command interpreter. The commands that you have access to are grouped into basic and advanced commands. In many cases, the basic commands suffice to spot the error you are looking for. For more intricate bugs, however, you can resort to the more advanced commands.

## Basic Commands

From a programmer's point of view, the basic commands allow for an enhanced *printf* facility with the option of switching the *printf* function on and off as well as stopping the execution at predefined stop points (breakpoints). A major asset is the ability to store and load the breakpoint configuration.

The question mark (?) activates the help comment, which will display all commands of the interpreter along with brief descriptions of their use.

The command *STOP n pattern* defines stop pattern *n*, where *n*=1, 2,…,10, which indicates a breakpoint. If the tool detects a *dbg()* statement with this pattern in the label part of *dbg()*, it will stop execution and prompt *DBG>>*. For Example 1(b), an appropriate command to reach the stop point would be *STOP 1 7_stp1*. Note that the parameters after STOP must be separated with blanks and that the parameter *pattern* does not store spaces; the label "*7_stp1* " (note the trailing space) will not trigger the *dbg()* function, as it cannot be represented properly by *pattern*. Always finish a stop label in *dbg()* with a nonwhitespace character.


There you are, struggling to develop your client/server application and enterprise-wide data warehouse under a tight time frame,
and then one day it hits you.
SQL Server
Java
C++
Oracle
Smalltalk
Sybase
Access
HTML
Booch
InterBase
Information Engineering
Gane/Sarson
OMT/Rumbaugh
Shlaer/Mellor
Ward & Mellor
PowerBuilder
Informix

## The System Architect Family of Tools

Just think, System Architect's integrated toolset can fulfill all your modeling needs. First, you can reverse engineer existing systems into SA's multi-user repository. Then analyze your system and business requirements using process and Use Case modeling. Continue with logical and physical data modeling for warehouses or data marts, and easily generate traditional and star schemas. Pretty bright idea wouldn't you say?

Well, that's just the beginning of SA's power.

When you're ready to develop decision-support software, you can use SA's OO tool to generate the code and the SA/4GL module to develop prototypes in PowerBuilder, VB and Delphi.

There's even SA/BPR, so the results of your business process reengineering project can be shared with your system design team.

Whether you're converting legacy systems to client/server environments, constructing data warehouses or using JAVA to build an inter/intranet application, System Architect customizable development tools can give you a clear competitive advantage... now, and far into the future. Who knows? It just might spark your next killer app, too.

To qualify for a free evaluation copy of SA,* please call us today at 800-732-5227, ext. 744, or fax us at 212-571-3436. We're also available at http://www.popkin.com.

800-732-5227, x744

POPKIN SOFTWARE & SYSTEMS, INC.

Real Tools For The Real World.

Visit us on the web at http://www.popkin.com

CIRCLE NO. 118 ON READER SERVICE CARD

*(continued from page 22)*

For *pattern*, you can also use the wildcards "?" and "*". For example, the pattern *7_s** will match all labels in your program starting with *7_s*. The pattern *7_stp??* will match any label starting with *7_stp* and having two additional arbitrary characters. Again, only the stop numbers 1 through 10 are available. However, this can be extended easily in the source code if you need more than ten stop points.

The GO command resumes execution and returns control to the program being tested. The program will run until the next stop point is encountered.

The command TRACE *n pattern* defines the trace pattern, where *n*=1, 2,...,10. If the tool detects a *dbg()* statement with this pattern, it will execute the *dbg()* statement; that is, it will print the desired message if the trace level allows it to. For Example 1(b), an appropriate command to achieve printout would be *TRACE 1 7_stp1*. The treatment of the parameters is the same as described for STOP. As with the stop command, only the trace numbers 1 through 10 are available. However, this can be extended if you need more than ten trace points. As a default, all trace labels are enabled, and the debugger acts as if *TRACE 1 ** were used at initialization.

The command LEVEL *n* specifies trace level *n* and suppresses all enabled trace points with a level parameter greater than *n*.

The STAT command displays the status of the debug/trace tool. The status comprises the activated stop and trace points, the trace level, and some advanced command parameters.

SAVE *filename* saves the status of the debug/trace tool in the file specified by *filename*. If just the command SAVE is used, STAT.DAT will be used as the default filename. The *save* feature is helpful if multiple runs of a program with the same trace, stops points, and the like are required, especially if the program code has been changed between runs. You can use this command not only for debugging, but also for regression testing, provided that your *dbg()* statements remain in the code. SAVE is used together with LOAD.

```
(a) void  dbg(char *stop, int
           level,char *format, ...);

(b) mean = 0;
    for  (j = min; j < max; j++)
    {
      mean = mean + array[j];

        dbg("7_stp", 2,
            "\n mean = /f",mean);
    }
    mean = mean / (max - min + 1);
```

***Example 1:** (a) Declaring* dbg()*;*
*(b) using the* dbg() *function.*

The LOAD *filename* command loads the status of the debug/trace tool from the file specified by *filename*. If just the command LOAD is used, STAT.DAT will be used as the default filename.

Finally, the command QUIT quits the debug/trace tool and finishes the program.

## Advanced Commands

Advanced commands extend the basic commands. For instance, you can skip the stop and trace points, store the trace output, and gain insight into the physical memory by using a dump and checksum facility.

*It sometimes helps to think of the code as divided into logical sections*

The GRAN *n* command sets the trace granularity to *n* so that only every *n*th *dbg()* call, the level variable of which is smaller or equal to the current trace level, will be printed. The argument *n* must always be greater than zero. GRAN *n* can be helpful when you know that a bug involves a certain variable. The variable ends up with a wrong value, but you are uncertain where or when the variable acquired that value. In such a case, you may decide to examine every thousandth reference to the suspect variable. Then, within the 1000 references that surround the error, you examine every hundredth reference. Finally, within the 100 references that surround the error, every reference is examined. This is referred to as an "adjustment in granularity."

The command SSKIP *n* is the stop-point skip command to skip *n* stop points. For example, by entering *SSKIP 3*, you tell the debug/trace tool to skip the next three activated stop points. The SSKIP command is particularly useful when you don't want to cancel the stop points (because you want to use them all again); just skip some in the present run to concentrate on a specific part of your program. A good example of this involves loops, where you just want to examine their beginning and end.

For instance, once you've examined the first output for u=0 and v=0 in Example 2, you might want to skip the next 34 stop points by entering *SSKIP 34* and resume with the last one. As an aside, once you've activated tracing on label *1_init1*, the output portion of the *dbg()* statement will always be executed, which means that 34 outputs are running over your screen. If you don't like this, you should first switch off tracing by entering LEVEL 1 before you type GO, thus changing the trace level. Alternatively, you can use the TSKIP *n* command, which is analogous to SSKIP *n* but has influence on the activated trace points.

The command TRCMD *c filename* specifies the trace mode, which selects either the console, the file specified by *filename*, or both as the output device. If no filename is specified, TRACE.DAT is the default. The options for *c* are: c, tracing to console only; f, tracing to file only; and a, tracing to console as well as file. The default setting for *c* is "console only." This is used every time the debugger is started and cannot be influenced by the LOAD command.

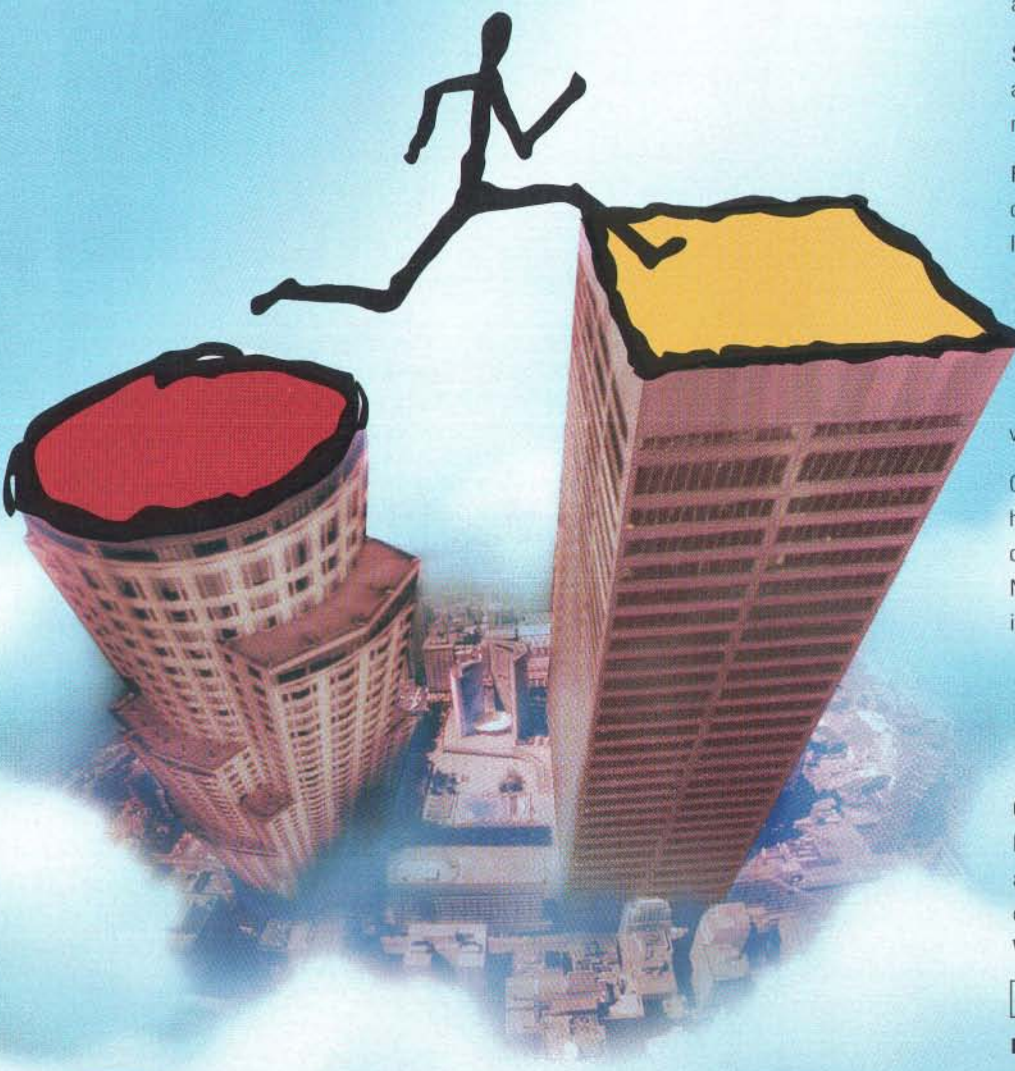
WATCH *n addr* defines a memory address that will be watched and assigns this address the identification number *n* (up to now, only 1, 2, and 3 have been implemented). Watch addresses are examined automatically at certain *dbg()* calls according to the watch mode selected. Whenever an examination reveals that the contents of a watch address have changed, a stop point is forced. The syntax of *addr* depends on the memory model used for 80x86 processors. For the tiny and small model, *addr* is a four-digit hexadecimal number. For the medium, compact, large, and huge models, *addr* consists of two four-digit hexadecimal numbers separated by a colon.

The command WATMD *c* specifies which *dbg()* calls trigger automatic watch examination. The options, specified by a single character, are: o, all watches off (the default); s, examine at stop points only; t, examine at trace points only; and a, examine at all occurrences of *dbg()*.

DUMP *len start* prints, in hexadecimal and ASCII format, the *n*-byte block of memory beginning at address *start*. The syntax of *start* depends on the memory model used for 80x86 processors (see WATCH *n addr*). If the parameters *len* and *start* are not specified (that is, only DUMP is used), the dump is performed using the previous values for *len* and *start*, if any.

The CRCK *len start* command computes a cyclic redundancy check over the *n*-byte block of memory, beginning at *start*. The notational remarks for *len* and *start* are the same as for DUMP *len start*. Checksums are a powerful technique for

# INTRODUCING YOUR NEXT STEP IN CLIENT/SERVER



Taking the next step in client/server can be tricky. You need larger, more scalable applications. But which architecture will you use? How will you keep up with emerging Internet technologies? How will you make it all work together?

Prepare for your next step with the new Centura™ family of products from Centura Software Corporation (formerly Gupta Corporation). Centura products are not only scalable, Internet-enabled, and open, but also flexible enough to support any application architecture.

**SCALE UP.** Tackle bigger, more demanding applications. Work with more developers, more databases, more users.

**REACH OUT.** Move beyond traditional corporate networks to embrace the Internet. Create applications that enhance your business.

**CONNECT IT ALL.** Unite disparate databases and integrate mobile users. Make all your data and applications work together.

Centura Software Corporation solutions have been proven by more than 100,000 developers and more than a million users. Now we're ready to help you step nimbly into the future of client/server.

**CENTURA**™
SOFTWARE CORPORATION
*formerly known as* **GUPTA**®

**CALL TODAY** to qualify for your **FREE** Next Step Evaluation Pack, including a demo and in-depth technical information. Or visit our World Wide Web site and register to **WIN** a copy of Centura Team Developer.

**1-800-444-8782 x407**

**http://www.centurasoft.com**

**CENTURA**

**1060 Marsh Road • Menlo Park, CA 94025 • (415)321-9500 • FAX:(415)321-5471**

*(continued from page 24)*
detecting errant pointers that write on code. The user simply recomputes the checksum at regular intervals during the program's execution. If the checksum at one computation differs from the previous computation, an intervening pointer reference must have modified the code. If the parameters *len* and *start* are not specified (that is, only CRCK is used), the checksum is computed using the previous values for *len* and *start*, if any.

```
for(u=0;u<6;u++)
 {
   for(v=0;v<6;v++)
    dbg("1_init1",2,
        "w[/d][/d] =
         /d   ",u,v,w[u][v]);
 }
```

***Example 2:*** *You can use the* SSKIP *command to skip the next 34 stop points.*

```
(a) #include <stdarg.h>
    extern void dbg_init(void);
    extern void dbg(char *, int,
                    char *, ...);

(b) void main(void)
    {
       int i, j;
       float x, y , z;
       dbg_init();
       ...
    }
```

***Example 3:*** *(a) Necessary debug statements in the program you want to debug; (b) initializing the debug/trace tool.*

## Additional Advanced Commands

Some commands are available only in basic form because their implementations depend either on the memory model, compiler, or application program. I've implemented these commands using the DOS small memory model. However, since the source code of the debugging facility is available and documented, you can adapt the appropriate functions to fit your environment.

STACK gives a stack dump of all stack locations with return addresses (frame-pointer chain).

The command STCMD *c* specifies which *dbg()* calls trigger automatic stack examination. The options, specified by a single character, are: o, stack examination off (the default); s, examine only at stop points; t, examine only at trace points; and a, examine at all occurrences of *dbg()*.

The LOCAL *n* command prints the local variables and parameters associated with the last *n* stack frames.

SNAP *n* invokes one of three user-defined snapshot functions (*n*=1,2,3). Snapshots usually are defined separately for each program tested and are used to dump global or local variables accessible via a globally known pointer.

## Using the Debug/Trace Tool

To use the debug/trace tool, you must include *dbg()* statements in your code wherever necessary. You must also include the statements in Example 3(a) in the program you want to debug. Next, you must place the function *dbg_init()* at the beginning of your program to initialize the tool, as in Example 3(b). Finally, you need to link the object file of the tool to the object file of the program under test. You can then run the program to be tested.

When instrumenting your program, it sometimes helps to think of the code as divided into logical sections. In the first section, all labels might start with *1_*, in the second, all labels start with *2_*, and so on. In each section, the trace level can be used to distinguish the nesting level, starting with level 1. Level 0 should be reserved so that tracing can be switched off if necessary. Listing One (listing begins on page 84) is C code that illustrates this approach.

**DDJ**

**(Listing begins on page 84.)**

Track Bugs Down

**Track All Bugs, Feature Requests, Fixes, Support Calls, and more...**

- **CUSTOMIZE:** Reports • Queries • Forms and Fields
- **NOTIFY:** MSMail • cc:Mail • Lotus Notes • More
- **INTEGRATE:** PVCS • SourceSafe • Source Integrity
- **USE:** dBaseIV • MS SQL Server • Oracle • More
- **ANALYZE:** Trends • Process • History • More

SOFFRONT® TRACK™

***Thousands of corporations worldwide have selected TRACK after evaluating their alternatives. Download the FREE evaluation copy ttoday!***

Win 3.1, 95, NT

"TRACK wins hands down over the competition in the flexibility, power and ease-of-use areas."— *DM Review, January 1996*

"The ability to get users up and running with TRACK in a very short period of time is invaluable." — *Greg Bryant, Principal Engineer, NEC Technologies, Inc.*

"This program is very flexible indeed."— *Windows Tech Journal, January 1995*

30-day, unconditional money-back guarantee

FREE Evaluation Copy!

Call Today!

**SOFFRONT**

**1-800-763-3766**

**SOFFRONT Software, Inc. Ph 408-263-2703**

238 S. Hillview Dr., Milpitas, CA 95035 • Fax 408-263-7452

**Internet: info@soffront.com http://www.soffront.com**

**CIRCLE NO. 318 ON READER SERVICE CARD**

**PCYACC® Version 6.0**

**PROFESSIONAL LANGUAGE DEVELOPMENT TOOLKIT**

***Includes "Drop In" Language Engines for SQL, dBASE, POSTSCRIPT, HYPERTALK, SMALLTALK-80, C++, C, PASCAL, PROLOG, FORTRAN, COBOL, BASIC, SGML, ASN, RPG, REXX, PL1, SNA, RTF, VISUAL BASIC, SQL2, DB2, VHDL, HTML, VMRL, JAVA, ODMG-ODL/OQL, SQL3, MODULA-3, DELPHI, VBS, and ADA.***

PCYACC Version 6.0 is a complete Language Development Environment that generates C, C++, Java, Delphi, and VBS source code from input Language Description Grammars for building Assemblers, Compilers, Interpreters, Browsers, Page Description Languages, Language Translators, Syntax Directed Editors, Language Validators, Natural Language Processors, Expert System Shells, and Query languages.

- **Portable Object Oriented Classes for C++ and JAVA – Error, Symbol Table, Syntax Tree, Yacc, and Lex.**
- **Debugging tools include runtime Visual Parser Debugging, Abstract Syntax Tree generation, and Cross Referencing.**

**CodeCheck® Version 6.0**

**SOURCE CODE ANALYST**

***Includes "Drop-In" Rules for Compliance analysis, Adherence to specifications, Measures of complexity, Silent error detection, Code maintainability, and Portability.***

CodeCheck Version 6.0 is a programmable tool for managing all C and C++ source code on a file or project basis. CodeCheck is input compatible with all variants of K&R, ANSI C and C++ (Microsoft, Metaware, Borland, Intel, Vax/Vms, HP/Apollo, Microtec, Watcom, Symantec, MPW, CodeWarrior, AT&T, and GNU). CodeCheck is designed to solve all of your Portability, Maintainability, Complexity, Reusability, Quality Assurance, Style Analysis, Library/Class Management, Code Review, Software Metric, Standards Adherence, and Corporate Compliance Problems.

- **Portability – CodeCheck identifies code that will not port between MSDOS, OS/2, Unix, VMS, WIN-95/NT, Macintosh and porting to 64 bit machines.**
- **Compliance – CodeCheck allows your corporate coding and project specification standards to be completely automated for compliance validation.**

CodeCheck includes pre-written expert system Rule Files that can be applied to any C or C++ project for: POSIX, SVID, X3J11, Ellemtel, and ANSI C/C++ Source Code Conformance Validation.

***30 day Money back guarantee! Free AIR Shipping anywhere in the world!***

To Order Call

**1-800-347-5214**

| | CodeCheck | PCYACC |
|---|---|---|
| DOS/WIN16 | $495 | $495 |
| MAC | $495 | $495 |
| OS/2 | $995 | $995 |
| WIN32 95/NT | $995 | $995 |
| UNIX/VMS | $1995 | $1995 |

**ABRAXAS™ Software, Inc.**

5530 SW Kelly Ave., Portland, OR 97201 USA
TEL (503) 244-5253 • FAX (503) 244-8375
E Mail: sales@abxsoft.com
We are on the web! http://www.abxsoft.com

**CIRCLE NO. 124 ON READER SERVICE CARD**

# Got a Bug? Find It. Fix It.

**Insure++**

ParaSoft

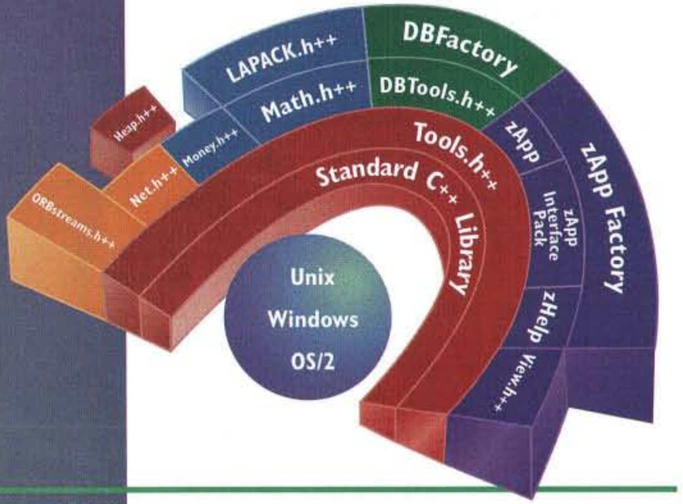
***Insure++*** **is the most thorough error detection tool available for C/C++ programmers. It will find not only your ordinary bugs, but also those hard-to-detect errors that cause major problems. It pinpoints memory leaks, memory corruption, dynamic memory errors, and much more.**

**Call (888) 305-0041 now to find out how you can start finding and fixing bugs today.**

*__Para__Soft Corporation, 2031 S. Myrtle Avenue, Monrovia, CA 91016*
*Phone - (818) 305-0041 • Toll Free - (888) 305-0041 • Fax - (818) 305-9048*
*Website - www.parasoft.com • E-mail - info@parasoft.com*

CIRCLE NO. 294 ON READER SERVICE CARD

# Build Your C++ Applications



Make sure your applications rest on a solid foundation by building them with software from Rogue Wave. You'll get reliability and portability, and you'll save time and effort by using (and reusing) our well-designed classes.

Base your applications on the data structures, collection classes, and algorithms found in Tools.h++ and the Standard C++ Library. Use zApp Developer's Suite to quickly build portable user interfaces with native look and feel. Tap into the wealth of classes and functionality in libraries such as Net.h++ and Math.h++.

Don't start your development in the basement! Rely on proven technology to get the job done. Rogue Wave's libraries and builders are portable among Unix, Windows, and OS/2, and full source code is available, with no royalties or run-time fees.

C++ Compiler Vendors Choose Rogue Wave!

Sun Microsystems

Hewlett Packard

DEC

SGI

Borland

Centerline

Metaware

Siemens-Nixdorf

Rogue Wave products are now available for purchasing and downloading at our web site.

## DBTools.h++

Version 2.1.1 available now!

### THE BENCHMARK FOR C++ DATABASE APPLICATIONS

**DBTools.h++** sets the pace by delivering performance that will satisfy even the DBTools.h++ users who measure their data sets in terabytes, and their transactions in millions per day. Regardless of the size of your database, you need C++ access without a lot of overhead. DBTools.h++ gives you the power and ease of use you want, and the performance you demand.

### POWERFUL NEW FEATURES

DBTools.h++ includes support for asynchronous database access. It also supports binding of variables to SQL statements to streamline processing of large data sets, and includes fully implemented multithreading capabilities (when supported by the compiler and database client library).

### YOU CHOOSE THE DATABASE, DBTOOLS.H++ DELIVERS THE C++ ACCESS

DBTools.h++ encapsulates SQL 92 DML functionality and delivers native access to Oracle, Sybase, Informix, Microsoft SQL Server, and Ingres databases, plus general connectivity through ODBC. Source code is available on Windows, Unix, and OS/2 platforms.

### CUT DBTOOLS.H++ DEVELOPMENT TIME BY 30% OR MORE!

**DBFactory** supercharges DBTools.h++ by automatically creating portable C++ objects that represent your database schema information. You control code generation through a point-and-click interface. Choose the elements you want to represent in your application, and DBFactory does the rest, creating .h files, .cpp files, and documentation for each generated class. DBFactory tracks your changes, so you can modify generated code today, regenerate to capture database changes tomorrow, and retain your modifications.

With DBTools.h++ and DBFactory, you'll write cleaner database applications, realize shorter development cycles, and reduce your maintenance costs, all without disruptive changes to your IS environment.

# On a Solid Foundation

## Tools.h++

### A World-Class Foundation

### Standard C++ Library

Build a solid foundation for your application with **Tools.h++**. Tools.h++ is an internationalized C++ foundation class library packed with over 120 reusable classes including sets, bags, sorted collections, strings, dates and times, extensible virtual streams for persistence, and more. Tools.h++ 7.0 adds endian streams, non-intrusive persistence, enhanced regular expressions, and a higher-level interface to the Standard C++ Library.

Rogue Wave's Standard C++ Library truly is the standard. Our role as a member of the ANSI/ISO committee allowed us to create a pure, conforming implementation of the new standard. Standard C++ Library includes updated data structure and algorithm classes, plus string, numeric limits, and complex classes, and allocators.

## Threads.h++

### Portable Multithreading

**Threads.h++** provides everything you need to add multithreading to existing applications or to write new multithreaded applications from scratch, including thread creation and control classes, thread synchronization mechanisms, and inter-thread communication features.

Use the Threads.h++ thread creation and control classes to create, interrupt, join with, and cancel threads; monitor their execution state; adjust their priority; and more! Use the library's complete set of synchronization classes to control access to shared resources. Represent the future result of an operation with the library's IOU feature.

Plus, when you use the Threads.h++ platform-independent programming model, you can write your multithreaded application once, then recompile to run in other environments.

## Inter.Net.h++

NEW!

### Simplifies Network Programming

### Net.h++

With the explosive growth in popularity of the Internet and corporate Intranets, more and more developers are incorporating TCP/IP-based protocols into a wide variety of applications. **Inter.Net.h++** offers C++ classes for the most popular Internet protocols, with a simple object-oriented interface. Plus, the classes are multithread hot! It includes support for client-side implementations of FTP, HTTP, POP3 and SMTP, and works in combination with **Net.h++** to improve your productivity and reduce the lines of code you write for your network applications. And, of course, the same code is portable between Windows and Unix!

**Net.h++** is a powerful C++ class library for developing network applications. It provides an easy-to-use object-oriented interface to TCP/IP, offering you a clean, maintainable way to write code for client/server and peer-to-peer communications. You can write directly to Net.h++'s portable C++ sockets abstraction, or you can use a higher level of abstraction that allows you to stream your data over a TCP/IP network using C++ iostream classes or Rogue Wave's virtual streams classes. The virtual streams classes provide a powerful persistence mechanism for Net.h++ applications. Net.h++ code is portable to all major Windows and Unix platforms, and to OS/2.

SOFTWARE Development PRODUCTIVITY AWARD 1995

zApp® Developer's Suite

Version 3.0 now available!

### zApp Developer's Suite 3.0

contains so many features and so much functionality that programming portable C++ GUIs is simpler than ever! The **zApp Factory** visual builder cuts your development time in half by allowing you to drag-and-drop predefined components from the tool palette to create your user interface, test the interface without compiling a single line of code, then generate fully commented C++ code with the click of a button.

#### FUNCTIONALITY END USERS DEMAND

The classes that ship with zApp Developer's Suite range from high-level controls such as tabbed dialogs and hierarchical list boxes to a highly sophisticated spreadsheet-like table object. They also include basic functionality such as size, positioning and drawing classes, as well as powerful geometry and clipboard classes.

#### ALL THE ZHELP YOU NEED

Add context-sensitive help to your applications in a snap with **zHelp**, our portable help library. zHelp covers all the bases from a single set of HTML source files. It provides full hypertext navigation of multiple help pages, including graphical links, and an automatic search engine. In addition to using preformatted text, you can control the look of your help pages with multiple fonts, italics, and bullets.

zApp Developer's Suite is now the 3-time winner of InfoWorld's product review

#### INDUSTRIAL-STRENGTH FOUNDATION

Best of all, version 3.0 of zApp Developer's Suite is integrated with Tools.h++, the C++ foundation class library relied upon by tens of thousands of developers worldwide.

## Professional Services

**On-site training** helps you develop your C++ expertise in a variety of topics, ranging from code design to specific instruction about Rogue Wave products. Our mentoring services take this transfer of expertise a step further, as we work directly with your development staff on projects.

The Software Parts Company™ Rogue Wave SOFTWARE

Surf Us http://www.roguewave.com → Call Us (800) 487-3217 or (541) 754-3010 → Fax Us (541) 757-6650 → Email Us sales@roguewave.com

CIRCLE NO. 215 ON READER SERVICE CARD

# Java GUI Testing

*Automated testing is as problematic as it is essential*

Alan Walworth

Graphical user interfaces may be a dream for users, but they can be a nightmare for testers. As Java development takes off, so does the need to test GUI applications written in Java. The newness of Java and Java-oriented testing tools makes this a challenge, however. This is as true for development groups within Sun (such as the WorkShop Products group, of which I'm a part), as for third-party developers. In this article, I'll examine Java GUI testing issues in general, then present some of the solutions we've adopted for testing Java GUI software.

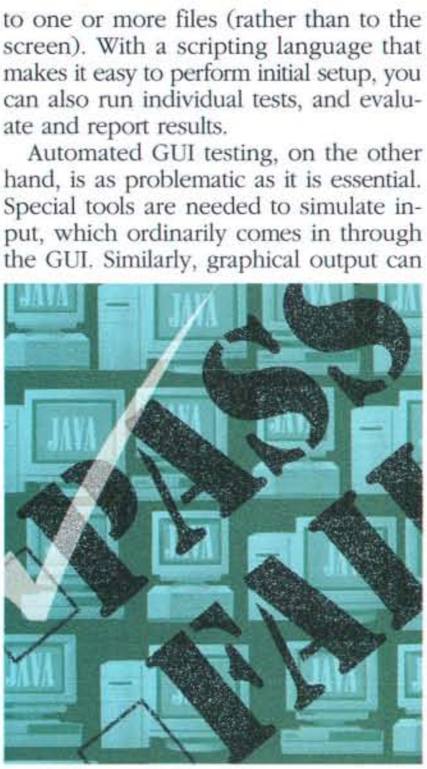
### Automated Testing

Though automated test suites can never entirely eliminate the need for manual testing, they are invaluable as an efficient way of:

- Detecting new problems quickly.
- Making sure old problems remain fixed.
- Verifying that a package satisfies fundamental quality criteria prior to shipment.

For applications with command-line interfaces, test automation can, as a rule, be readily accomplished with the help of I/O redirection, which feeds the application input from a file (rather than from the keyboard) and sends the application's output to one or more files (rather than to the screen). With a scripting language that makes it easy to perform initial setup, you can also run individual tests, and evaluate and report results.

Automated GUI testing, on the other hand, is as problematic as it is essential. Special tools are needed to simulate input, which ordinarily comes in through the GUI. Similarly, graphical output can



be captured and analyzed only with special-purpose tools. For Java GUI applications, the need to use special testing tools is unusually problematic — because of Java's newness, the necessary tools are just beginning to appear.

Since a major attraction of Java is its platform independence, testing Java applications (or applets) tends to require testing on multiple platforms. On the face of it, therefore, the tools need to work on all targeted platforms. A colleague facetiously suggested that there is really no need for multiplatform tools for Java testing, because a Java program verified to work properly on one platform is guaranteed to work identically on all others. "Write once, run anywhere" is indeed one of the promises of Java, but remember that Java is still a relatively immature language and is by no means perfect. Early versions of the AWT (Abstract Window Toolkit), in particular, behave differently (and at times quite improperly) depending on the operating system used. Presumably, these problems will be significantly diminished in JDK 1.1, but for the foreseeable future, serious testing of Java applications will require testing on all major platforms.

There are two basic ways of creating the required sort of multiplatform Java GUI test tool. The first option, adding Java capability to a preexisting non-Java tool, has the advantage of inheriting the strengths of the preexisting tool. On the other hand, such a tool is not guaranteed to run on all platforms. The second option, of course, is to create a Java-based tool from scratch that can be expected to migrate automatically to new platforms along with the application being tested. In the short run, however, the new software is likely to lack refinement. Our search for test-development tools turned up one candidate of each kind — Segue's QA Partner and SunTest's JavaSTAR.

### QA Partner

The most recent version of Segue's QA Partner (http://www.segue.com/), a well-established GUI testing tool running on UNIX, Windows, OS/2 and Macintosh, incorporates a "Web-aware" capability which, among other merits, provides the ability to test Java applets and applications. Given its non-Java roots, QA Partner naturally achieves this capability not by interacting with AWT components directly, but by interacting with associated native GUI components on the various platforms, such as Motif widgets on UNIX and Windows controls on Windows 95/NT.

For Java testing, this strategy has pros and cons that need to be considered according to your needs. Basing testing on native components is attractive for testing applets running under a browser such

*Alan, a software-quality engineer for Sun's WorkShop Products unit, can be contacted at alan.walworth@Eng.Sun.COM.*

If you're not creating Internet applications on the Internet,
you're someplace you shouldn't be.

Java development tools from Sun.

Leave it to the creators of the Java™ platform to give you Java™ WorkShop.™ The only way to create and test Internet applications where they'll run, on the Internet. Making it easier and faster to develop Java applications. Plus it's written in Java, so you can develop on Solaris™ O/S, Windows 95, Windows NT and soon on the Mac OS. $99* gets you Java WorkShop and all the upgrades for a year. To find out more contact 1-800-SUNSOFT or www.sun.com/developer-products/ Develop, deploy & manage with SunSoft WorkShop,™ Solaris™ & Solstice.™ **THE NETWORK IS THE COMPUTER™**

Sun microsystems

**CIRCLE NO. 257 ON READER SERVICE CARD**

*(continued from page 30)*
as Netscape Navigator, which itself uses the native GUI. With a test tool that uses native components, Netscape Navigator events can be captured, and Navigator interface properties tested, along with the events and interface properties of the Java applet (or more precisely, along with those of the native implementation of the applet's GUI elements).

On the other hand, the tool is not directly aware of the Java components, specifically at the AWT level. Therefore trouble may result from inability to recognize Java (AWT) component events and properties, or from inability of a single test script to cope with the differing native manifestations of the same Java component from one platform to another.

For example, the JDK 1.0.2 AWT *FileDialog* has different native implementations on Windows and UNIX, each containing native components (such as text fields) that do not match up with corresponding components at the AWT level. The AWT interface allows interaction with the *FileDialog* as a whole, rather than with particular items (such as native text fields) within the *FileDialog* implementation.

Thus, if a tool that is aware of only the native components is used to record interaction with a *FileDialog* on UNIX, the resulting test script will not be able to handle the corresponding interaction on Windows, since the Windows components will differ. Consequently, at least the *FileDialog* interaction will have to be rerecorded. For a tool that operates at the AWT level, on the other hand, the details of the native components from which the *FileDialog* is constructed are ignored, so the differences in native implementations will not prevent a test script recorded on one platform from playing back unaltered on another.

Because we primarily wanted to test applications such as Java WorkShop which are written entirely in Java, the ability to handle Netscape Navigator or other native browser events was not a significant benefit in our situation. (Though Java WorkShop is browser based, it uses an adaptation of the HotJava browser, which is written in Java.) Also, in our particular case, QA Partner's lack of Macintosh Java support was a hindrance. (Segue has expressed interest in implementing Macintosh Java capabilities if there is sufficient demand.)

### JavaSTAR

JavaSTAR is a Java GUI test tool from SunTest, a Sun unit that develops Java testing tools; see http://www.suntest.com/. As a purely Java-oriented tool, it is aware of AWT components, recognizing them the same way regardless of platform. The corresponding drawback, of course, is that JavaSTAR is unaware of non-Java components and thus cannot be used to directly control or test them. Again, this was not a problem for us because our target application (Java WorkShop) was written entirely in Java.

However, Java WorkShop's installation and startup are not Java based; on Solaris, for instance, installation uses tar. Windows installation employs InstallShield. Thus JavaSTAR cannot help us test Java WorkShop installation, except indirectly by determining whether Java WorkShop works after it has been installed. Since installation is relatively straightforward, this limitation was not a serious concern.

*Testing Java applications (or applets) tends to require testing on multiple platforms*

Platform-specific startup mechanisms were a more serious issue. On UNIX, for example, Java WorkShop uses a wrapper script to arrange the proper execution environment; on Windows, it uses an executable (jws.exe). Though details of getting the environment right — setting paths and the like — tend to differ from one platform to another, you may wonder why the housekeeping cannot, nevertheless, be done with Java, since a Java program can determine which operating system it is running on (with *System.getProperty("os.name");*) and should therefore be able to tailor the setup accordingly. However, there would still be a residual problem of invocation complexity. On Windows, for instance, users should not be forced to type a relatively complicated command such as "java sun.jws.Main"; plain "jws" is obviously preferable. Although platform-dependent configuration can be handled by Java, existence of at least a minimal wrapper script or executable is, at least for the time being, unavoidable. Given that there must be a wrapper in any event, it is hard to make a compelling case that the tool must handle configuration elsewhere. Though we lacked a way to involve startup wrappers in automated test runs, we determined that we could nevertheless keep JavaSTAR tests fully platform independent by reimplementing the necessary path setting operations in a Java class that could be called from the test harness.

Another factor in favor of JavaSTAR was that, because it was written in Java, we should be able to use it on any platform on which Java WorkShop might be supported in the future. To be sure, there will be new startup issues to resolve in moving to additional platforms, but these problems shouldn't be insurmountable.

On the other hand, an obvious drawback of JavaSTAR (when we began evaluating it in August, 1996) was that it was new, and in many respects, rudimentary. For example, paths to golden directories were generated using forward slashes, with the consequence that they worked only on UNIX, not on Windows. However, such glitches were fixed with successive releases.

### Test Harnesses

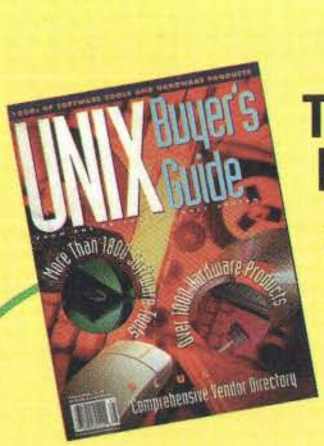
Being able to construct a GUI test that runs without human intervention is a big step. However, automated testing requires running collections of tests, with convenient mechanisms for controlling the set of tests to be run and for controlling the conditions under which the tests are run (such as the set of binaries used). Consequently, a test "harness" is required to manage the tests. I won't go into the ins and outs of test harnesses. Instead, I refer you to a minimal harness I wrote that provides basic functionality (available electronically; see "Availability," page 3). This Java test harness is a work-in-progress and is provided for illustrative purposes only.

Having decided to use JavaSTAR to construct our tests, our next challenge was to choose a "real" harness. The only one we found for running Java tests was JavaTest from JavaSoft with its Java API test suite (also known as the Java Compatibility Kit, or JCK) harness. At the outset, JavaTest presented two advantages. First, it had been in use for some time and thus was relatively mature. Second, we had already used it as a framework for a non-GUI benchmark test suite we had constructed, and we wanted to avoid a proliferation of harnesses that would complicate the job of running any arbitrary subset of our test suites. Another advantage was that JavaTest was written in Java and thus could be expected to run on all platforms.

However, during our initial review in August 1996, JavaTest was deficient in its ability to manage GUI tests. For instance, it was designed to manage suites of API tests, a task that differs in important respects from managing GUI test suites. A typical API test is a short program, perhaps 20 lines, written with the specific aim of testing a particular API feature. JavaTest assumes that the test developer has

What single source has more Servers than the French Open, more BASICs than elementary school, more RISCs than the Stock market, and more CASEs than the Supreme Court

?


UNIX Buyer's Guide
More Than 1800 Software Tools
Over 1000 Hardware Products
Comprehensive Vendor Directory

**The UNIX Buyer's Guide.**

Now available on the world wide web, complete with thousands of product listings, and a free-text and fielded search engine.

http://www.unixreview.com

*(continued from page 32)*
total control over the test program, and requires that the test program interface with the harness in specified ways, the most straightforward of which is that the test program's "main" method must return one of a limited set of test-status indicators (PASSED, FAILED, NOT_RUN, and so on). On the face of it, this won't work when the test program is large, preexisting, and not under the test creator's control.

You can envision getting around this restriction by arranging for the program JavaTest regards as the test to be a wrapper program that conforms to the JavaTest interface requirements and runs the application you actually want to test. However, this approach does not solve the fundamental problem that only meager results are passed back to the harness, with the result that, on the face of it, the harness will be able to generate only a limited report of test results. Clearly it is not enough simply to run tests automatically; reports that make it reasonably easy to pinpoint the causes of failures are also essential. When the test consists of just a few lines of code designed to exercise a single limited bit of functionality, a simple PASS or FAIL report is fine. A GUI test that brings up a large, complex application and exercises it with a stream of inputs to test a particular condition, on the other hand, has many potential points of failure. Consequently, the report of a failing test needs to indicate where in the elaborate sequence of procedures the trouble occurred.

A closely related issue is subtest-handling capability. The overhead of bringing up a large application such as Java WorkShop and manipulating it into a particular state for performing a set of tests makes it highly undesirable to restart that application for each individual test. The obvious solution is to bring it up once per test, but to conduct several subtests within that test. In other words, as we exercise the application, we'll check that various conditions are satisfied along the way—for instance, we may want to check that after a given button is pressed, a certain dialog pops up, with a field filled in with a certain string. If the dialog does appear properly, we may then want to exercise it and check further results. If we conduct enough subtests as we exercise our application, we may not need much detail in reporting results for each subtest. The nature of the subtest and our knowledge of preceding subtest results should give us a pretty good idea what went wrong. Unfortunately, just as API-oriented harnesses such as JavaTest fail to allow for detailed reporting of results of each test, they also tend to fail to report the results of subtests within an elaborate test. This is not to say that these harnesses are faulty, but that test developers need to recognize that harnesses designed for API testing (or for any testing based on running small programs written solely to test a particular limited feature), cannot be expected to have the special capabilities needed for large-scale GUI testing.

Additional features we wanted, but that JavaTest lacked, included filtering capability and reporting of golden-file differences. We would like, for example, to create an interface with the Java WorkShop GUI builder, generate source code for the interface, and then compare the generated source with previously generated source that has been established to be correct, or "golden." We found that although JavaTest allowed for golden-file comparisons, the resulting report merely indicated whether the comparison succeeded or failed, not how the actual output differed from the golden version.

Filtering is needed when a result (such as the content of a text field) includes data that varies depending on the conditions under which the test is run. The obvious example is the full path of a file; for example, a file storing information about the project currently active in a development environment being tested. The first part of such a path will differ depending on where in the file system the test is being run. In comparing such a result with the expected value, we need to remove the variable part before doing the comparison to avoid wasting time with false failure reports. Again, this sort of capability was not a priority for the API testing for which JavaTest was designed, so JavaTest did not provide it.

## JavaHarness

Because JavaTest did not possess the needed GUI capabilities, I began work on an alternative harness designed with GUI testing in mind. This effort ultimately resulted in development of "JavaHarness," a general-purpose harness for Java testing that SunTest makes available with JavaSTAR and JavaScope (a Java coverage tool). JavaHarness is, of course, written in Java, so it will run wherever needed for Java testing. Because it was initially designed for GUI testing, JavaHarness naturally addresses some of the key requirements neglected by JavaTest. JavaHarness reports subtest results, reports how golden-file comparisons failed, and provides filtering capabilities.

For the WorkShop Products group, an important attraction of JavaHarness is that it enables us to use a single test harness to run all our test suites, including those already written for JavaTest. JavaHarness can function as a meta-harness as well as a harness, allowing it to start up a JavaTest test suite instead of, or in addition to, one or more JavaSTAR test suites. The main JavaHarness configuration file, JavaHarnessSetup.ini, can include an essentially unlimited number of sections, each containing specifications for running a particular test suite. For a JavaSTAR test suite, the specifications refer to a separate file containing a list of tests to be run. For a JavaTest test suite, the specifications point to the ".jtp" file containing the information JavaTest needs to set up and run a set of tests.

The all-too-familiar pattern with test harnesses is that each is developed with a particular sort of testing need in mind, limiting its usefulness for other sorts of testing. JavaHarness attempts to be more flexible, providing interfaces that let users tailor its capabilities to a particular kind of testing. For instance, when API testers wanted to try using JavaHarness to run their tests directly (rather than via JavaTest), it was easy to create a new JavaHarness test type, *CompileAndRun*. The harness, when given a list of source files, then compiles and runs each in turn, reporting how each test program's output compares with the corresponding golden file.

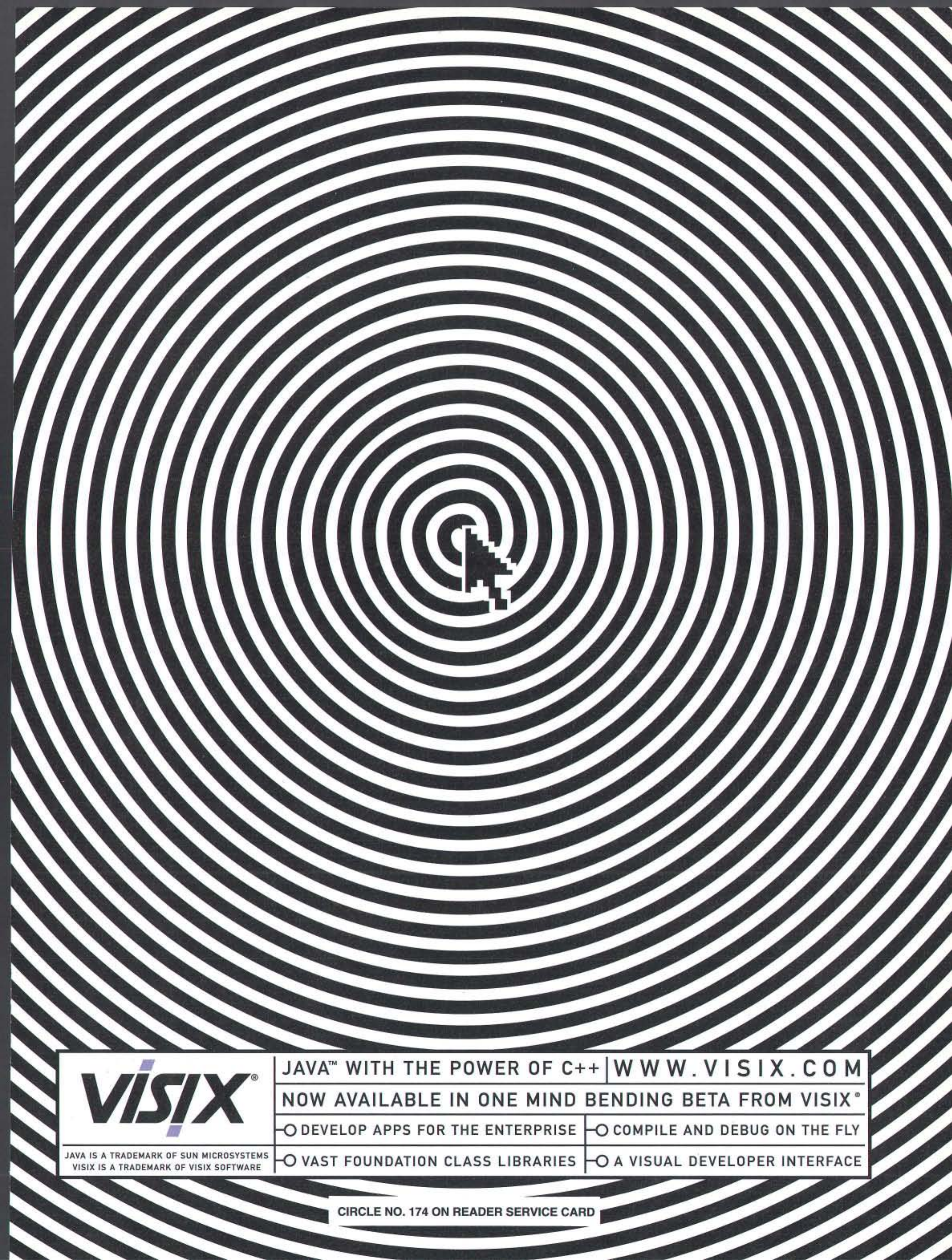
The chief drawback of JavaHarness, at this writing, is that it is still in an early stage of development. Though it provides the most essential harness capabilities needed to get a serious Java GUI testing program off the ground, we look forward to considerable refinement.

## Conclusion

Though problematic, automated testing is essential to guarantee high quality in substantial Java applications with complex GUIs. Though tools to help automate Java GUI testing are still immature and will doubtless become much more attractive with future refinement, they can already provide significant benefit.

In choosing tools for Java GUI testing, it is important to consider whether testing will be confined solely to Java GUIs or whether there will be a mix of Java and non-Java user interfaces. If the testing must involve the use of non-Java interfaces, as in testing the running of applets under the control of a browser such as Netscape Navigator or Microsoft Explorer (which are not written in Java), have a look at Segue's QA Partner. For testing purely Java applications (or applets with Java-based browsers or players such as HotJava or AppletViewer), tools written in Java are preferable due both to their direct awareness of AWT components and to the fact that their portability will automatically match that of the application being tested.

**DDJ**



VISIX®

JAVA™ WITH THE POWER OF C++ | WWW.VISIX.COM

NOW AVAILABLE IN ONE MIND BENDING BETA FROM VISIX®

- DEVELOP APPS FOR THE ENTERPRISE
- COMPILE AND DEBUG ON THE FLY
- VAST FOUNDATION CLASS LIBRARIES
- A VISUAL DEVELOPER INTERFACE

CIRCLE NO. 174 ON READER SERVICE CARD

# A Disassembler Written in Perl

*Using Perl's built-in pattern-matching capabilities*

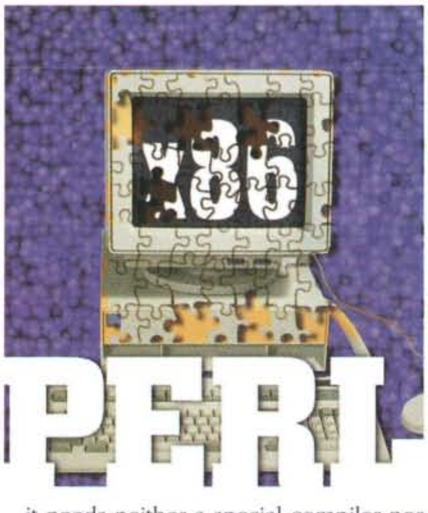
Tony Zhang

Reverse engineering has many applications, including disassembly and microprocessor debugging. Disassembly can be helpful in understanding code written by other programmers. In this article, I'll present the core subroutines of a disassembler written in Perl. This disassembler core is designed for the Intel x86 instruction set. You can modify or customize the disassembler for your own applications. To do so, simply design your application and write a low-level function to read assembled machine code (which may be from a static .EXE file or a bus trace fetched at run time). You then call the core disassembler routines to translate the assembled machine code (that is, the .EXE code) back into human-readable code, which should be close enough to the original source code for most purposes. For instance, I built a caller routine based on bus trace data collected by a logic analyzer connected to a PC system. The caller extracts the executive code from the bus traces and disassembles it by calling the core disassembler routines presented here.

*Tony, a systems software developer at Texas Instruments, can be contacted at tt-zhang@ti.com.*

There are several reasons for writing the disassembler in Perl:

- Perl has rich built-in features to easily manipulate text, files, and processes.
- Perl is ideal for providing quick solutions to programming problems because it needs neither a special compiler nor a linker to convert the source code to machine code.
- Perl is rapidly becoming a popular scripting language across almost all platforms.



## Instruction Format

Figure 1 shows the general format of the x86 instruction set. All instruction encodings are subsets of the general format. An instruction consists of four main parts. The first part is an optional prefix. The second part—the opcode (operation code)—is the most important and is one or two bytes long. The third part is an address specifier consisting of the ModR/M and scale index base (SIB) bytes, also optional. The last is for displacement or immediate data, if required.

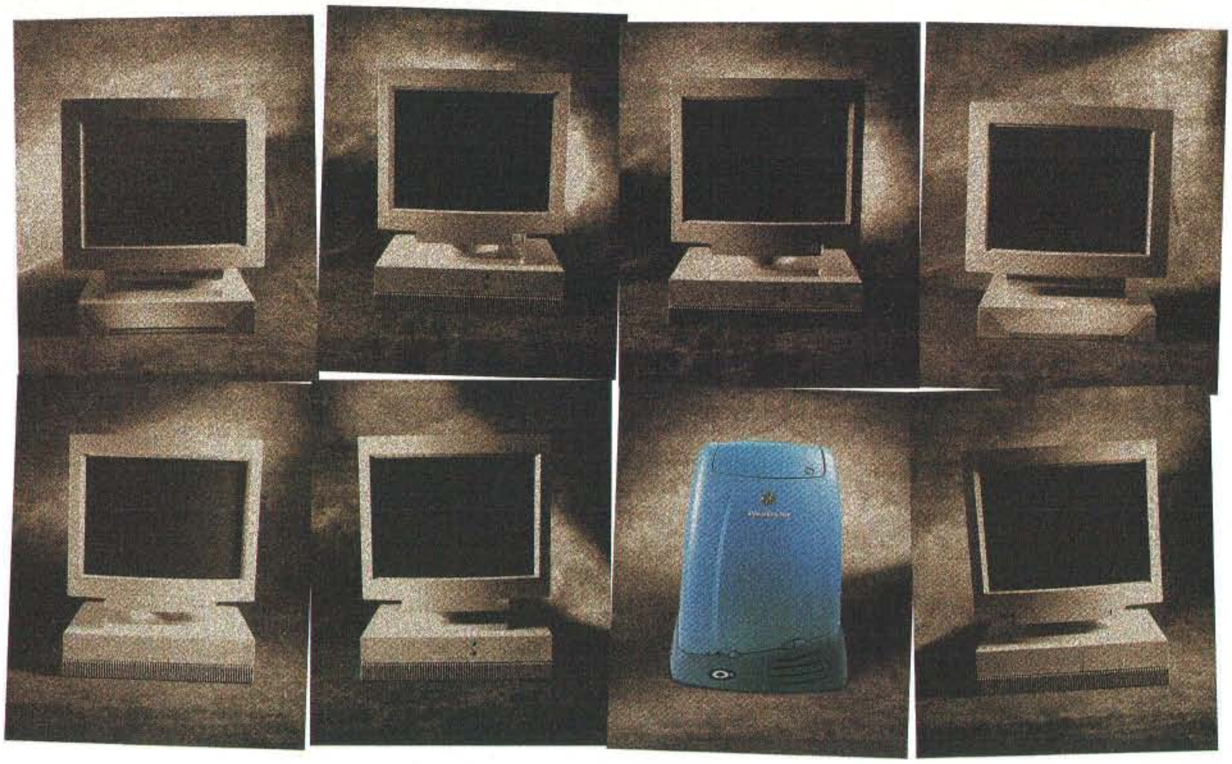
## Construct Instruction Look-Up Table

The file disasm.lut (available electronically; see "Availability," page 3) is an instruction look-up table based on the opcode map of the x86 instruction set given by Intel (see *Pentium Processor Family Developer's Manual, Volume 3: Architecture and Programming Manual*, Intel 1995). The ":" is used to separate the instructions and operand types. The number value of the first column indicates whether it is a 1-byte opcode map, a 2-byte opcode map, or another map determined by some attribute.

Each operand follows a "%" sign for parsing purposes. There are several windows in 1-byte and 2-byte opcode maps, such as *%g0* and *%g[a-q]*, that will link to 2-byte or other code maps.

A subroutine called "MakeLUT" is called to read the instruction look-up table into three arrays in memory when the disassembler is run for the first time. The three arrays are *@One_Byte_LUT*, *@Two_Byte_LUT*, and *%Three_Bit_LUT*. The last one, *%Three_Bit_LUT*, is an associative array, indexed by a string, *$lsTemp_1*, in the subroutine MakeLUT. The associative array is one of the most important features in Perl because it can be used for efficient database manipulation. It's often said that you are not thinking in Perl until you start thinking in terms of associative arrays.

The second column in the instruction look-up table gives the start index to the arrays of *@One_Byte_LUT* and *@Two_Byte_LUT*. For the associative array *%Three_Bit_LUT*, the combination of the second and third columns is used as the index to it. For instance, when the string *1:94:xchg %eax, %esp:xchg %eax, %ebp:xchg %eax,*



WE FINALLY MADE A WORKSTATION

# just like everyone else's.

We finally made a workstation just as practical, reliable, and affordable as everyone else's. With one minor exception: it's better. Introducing O2™. The only workstation that can combine industry-leading CPU and graphics performance with breakthrough video and imaging capabilities. Why? Because O2 is the only workstation based on an innovative Unified Memory Architecture. O2 comes standard with the MIPS® R5000™ chip, and is also available with the much more powerful MIPS® R10000™ CPU. Best of all, O2 is designed to be an extrovert. Every O2 machine comes standard with a full set of web-authoring tools as well as a personal web server. So as soon as you plug it in, you can communicate your ideas to anyone, anywhere, on any computer. If you want the performance of a workstation combined with the power of the web, look for O2. It isn't hard to find. For more information, visit our Web site or call 800.636.8184 Dept. LS0055.

**O2 Desktop Workstation**
**$7,495**
**MIPS R5000 180MHz processor**
**32-bit double-buffered graphics**
**Hardware texture mapping**
**Image processing engine**
**Video compression engine**
**Web-integrated user environment**
**64MB ECC SDRAM**
**2GB SCSI system disk**
**17" monitor, 1280x1040**
**100BaseTX/10BaseT Ethernet**
**CD-ROM**

SiliconGraphics®
Computer Systems

See what's possible

www.sgi.com/O2

CIRCLE NO. 164 ON READER SERVICE CARD

*(continued from page 36)*

*%esi:xchg %eax, %edi* is read in, MakeLUT first determines that the string should go to the array *@One_Byte_LUT*, because the number in the first *column(field)* is 1. The number 94 will be used as the start index to the array. Therefore, it will look something like Example 1(a) in the array *@One_Byte_LUT* after the reading.

Similarly, after reading the string *3:b:0:add %Ev, %Iv:or %Ev, %Iv:adc %Ev, %Iv:sbb %Ev, %Iv,* the associative array *%Three_Bit_LUT* will have four more items, as in Example 1(b).

## Parse Instruction

The main routine of the disassembler core, *X86_Disasm*, accepts a string containing the name of a low-level function that will be passed to subroutine *ImageByteRead* to read in image data (that is, assembled machine code). Also, the start address (either a linear or physical address) and the operand size (16 or 32 bit) will be passed to *X86_Disasm*. An instruction that accesses words (16 bit) or double words (32 bit) has an operand size attribute of either 16 or 32 bits.

When one byte of machine code is fetched by calling *ImageByteRead*, it is used as an index to find a string of symbolic instruction from array *@One_Byte_LUT*. Then a *while* loop is applied in order to find prefixes in the symbolic string (see Figure 1). Perl provides two operators to check pattern matching. In the *while* loop, you check whether the *$lsOpcode* contains *%P* or not by using one of the patterns matching operator *$lsOpcode =~ /\%P.*/*.

The subroutine *MethodP* contains all four possible prefix groups:

- Instruction Prefixes: *rep, repe/repz, repne/repnz*, and *lock* (where *rep* and *repe/repz* share the same cell in the opcode map).
- Segment Override Prefixes: *cs, ds, es, fs, gs*, and *ss*.
- Operand Size Override (*$DisasmnOS*).
- Address Size Override (*$DisasmnAS*).

According to Intel, most instructions that can refer to an operand in memory have an addressing-form byte (the ModR/M byte) following the primary opcode byte(s). The ModR/M byte specifies what kind of address form is to be used. Also, in certain cases, the ModR/M byte is followed by a second addressing byte, called the SIB byte, that is required to fully specify the addressing form.

The disassembler I present here (available electronically; see "Availability," page 3) has three subroutines—*ModBits, RegBits*, and *RMBits*. These check the three fields of *mod, reg*, and *r/m* in the ModR/M byte. *SIBField* returns the displacement and the register number of the index register, or the base register, according to the format of the SIB byte. Figure 2 illustrates the formats of the ModR/M and SIB bytes.

After the prefix-adding *while* loop, the symbolic string is parsed further to find if there are any links to arrays *@Two_Byte_LUT* or *%Three_Bit_LUT*. The pattern *%g0* implies that a new string of symbolic instruction will be needed from *@Two_Byte_LUT*, while *%g[a-q]* leads to the associative array *%Three_Bit_LUT*, according to how you built your look-up table.

Later, each character in the symbolic string is extracted and checked in order to find the corresponding opcode, necessary information regarding the ModR/M byte, the SIB byte, displacement, or immediate data. The Perl *substr* function is applied to extract those characters from the symbolic string. The tested characters will help to determine which subroutines should be called to interpret the opcode following the addressing methods.

The subroutines in the sample disassembler complete the job of translating the symbolic string into text-like assembly code for X86 instructions.

- *MethodA* represents the direct address in which the instruction has no ModR/M byte except the encoded address of the operand. Therefore, no base register, index register, or scaling factor can be applied in the addressing method.
- While *MethodC* uses the *reg* field of the ModR/M byte to select a control register, *MethodD* uses the *reg* field value to determine a debug register.
- *MethodE* fetches the ModR/M byte that follows the opcode and specifies the operand. The operand can be either a general register or a memory address. If the operand is a memory address, it is calculated from a segment register and any of the following values: a base register, an index register, a scaling factor, or a displacement. Following the calculated address, the subroutine can obtain all operand codes, convert them into proper formats, and save the results into a string variable that contains the final disassembled code.
- *MethodM* is basically the same as *MethodE*, except that *MethodM* is used only for a memory address. Similarly, *MethodR* is the same as *MethodE*, except that *MethodR* is used only for a general register.

```
(a)  $One_Byte_LUT[ 94 ] = "xchg  /eax, /esp";
     $One_Byte_LUT[ 95 ] = "xchg  /eax, /ebp";
     $One_Byte_LUT[ 96 ] = "xchg  /eax, /esi";
     $One_Byte_LUT[ 97 ] = "xchg  /eax, /edi".

(b)  $Three_Bit_Lut{ b:0 } = "add  /Ev, /Iv";
     $Three_Bit_Lut{ b:1 } = "or   /Ev, /Iv";
     $Three_Bit_Lut{ b:2 } = "adc  /Ev, /Iv";
     $Three_Bit_Lut{ b:3 } = "sbb  /Ev, /Iv".
```

***Example 1:*** *(a) Reading the array* @One_Byte_LUT; *(b) reading the array* %Three_Bit_LUT.

| Prefixes | Opcode | ModR/M | SIB | Displacement/Immediate |
|---|---|---|---|---|

| | | |
|---|---|---|
| Prefixes: | Instruction Prefix | 0 or 1 byte |
| | Address-size Prefix | 0 or 1 byte |
| | Operand-size Prefix | 0 or 1 byte |
| | Segment Override | 0 or 1 byte |
| Opcode: | | 1 or 2 byte(s) |
| ModR/M: | | 0 or 1 byte |
| SIB: | | 0 or 1 byte |
| Displacement: | | 0, 1, 2, or 4 byte(s) |
| Immediate: | | 0, 1, 2, or 4 byte(s) |

***Figure 1:*** *Instruction format.*

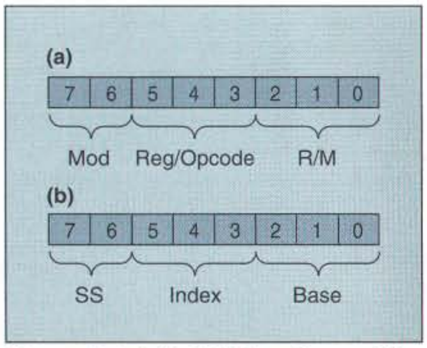
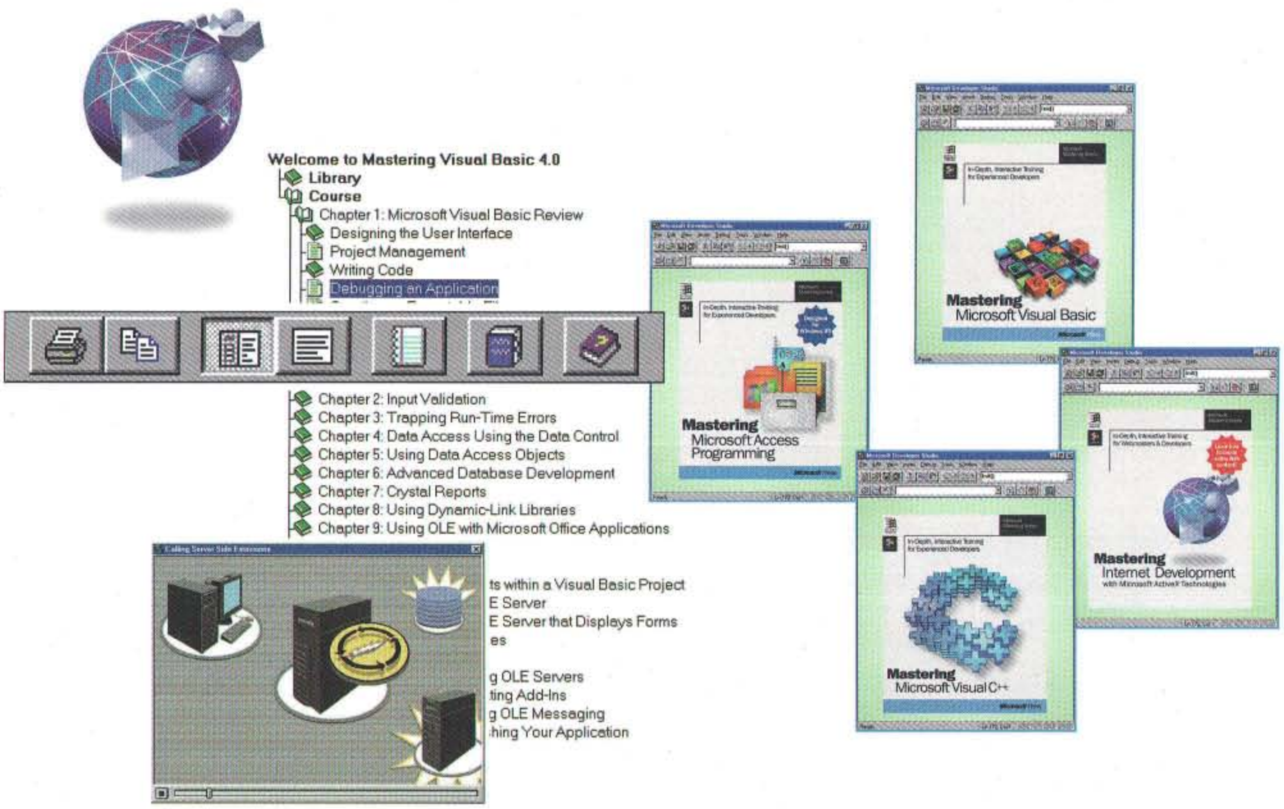


***Figure 2:*** *(a) ModR/M byte format; (b) SIB byte format.*

# Microsoft Mastering Series

**teaches you what you want to learn. Without waiting for the rest of the class to catch up.**



Welcome to Mastering Visual Basic 4.0
Library
Course
Chapter 1: Microsoft Visual Basic Review
Designing the User Interface
Project Management
Writing Code
Debugging an Application
Chapter 2: Input Validation
Chapter 3: Trapping Run-Time Errors
Chapter 4: Data Access Using the Data Control
Chapter 5: Using Data Access Objects
Chapter 6: Advanced Database Development
Chapter 7: Crystal Reports
Chapter 8: Using Dynamic-Link Libraries
Chapter 9: Using OLE with Microsoft Office Applications
ts within a Visual Basic Project
E Server
E Server that Displays Forms
es
g OLE Servers
ting Add-Ins
g OLE Messaging
hing Your Application

Mastering Microsoft Access Programming
Mastering Microsoft Visual Basic
Mastering Internet Development with Microsoft ActiveX Technologies
Mastering Microsoft Visual C++

**Every developer wants to be on the cutting edge. After all, it's what makes you so valuable. But not everyone has the time or money to attend an instructor led class. With Microsoft® Mastering Series CD titles, you'll learn exactly the skills you want, the way you want. Without the huge financial or time commitment. Best of all, class takes place in the world where you really work–your own desk. With Mastering Series' self-paced training for developers, you can read the course, do labs, see narrated demos, use sample code, reference articles and more. Jump ahead, switch sections, or go straight to the information you need to solve immediate problems. It's all according to your style. Mastering Series CD titles come straight from the source, the experts who actually built the tools and technologies you use every day. So it's based on real-life scenarios, not product features. There's really no other training of its kind, and no better way to stay ahead of the class.**

**Microsoft®**

**Where do you want to go today?® www.microsoft.com/mastering/**

*(continued from page 38)*

- A general register can be selected by calling *MethodG*, which checks the value of the *reg* field in the ModR/M byte. For the immediate data encoded in subsequent bytes of the instruction, *MethodI* can be used. *MethodJ* computes the relative offset in an instruction and adds the offset to the instruction pointer.
- For an instruction that has no ModR/M byte, the offset of the operand can be determined by calling *MethodO*. In this case, the offset is encoded as a word or double word in the instruction. The address-size attribute, *$DisasmnAS*, can tell whether it is a word or double word.
- Finally, *MethodS* selects a segment register by checking the value of the *reg* field in the ModR/M byte.

There are three variable scopes in Perl—global (the default), local, and my. The life span of the global is as long as that of the whole program module. A local variable in a calling routine can extend its life to a called subroutine. A my variable is limited to the scope of a routine itself. *$DisasmnPrefix*, *$DisasmnOp*, *$DisasmnModRM*, *$DisasmnSIB*, *$DisasmnDispl*, *$DisasmnImmed*, and *$DisasmnMapIndex* are defined as local variables so that they can be updated and passed easily among different subroutines.

Currently, *ImageByteRead* returns only one byte at a time. The string *$mem_read* in ImageByteRead contains the name of a low-level function that reads an image file (assembled machine code) and returns data in bytes. The integer *$byte_num* determines how many bytes of data should be read and returned.

(a)
```
($i=$lnTemp; $i<$lnTemp + $DisasmnDispl; $i++) {
  $lsResult = sprintf ("/02x",
      &ImageByte( $sMemRead, $i, 1 ) ) . $lsResult;
}
```
(b)
```
$lsResult = sprintf ("/x",
      &ImageByte( $sMemRead, $lnTemp, $DisasmnDispl
```

***Example 2:** The code in (b) is a rewrite of (a).*

## Building a Sample Disassembler

Disasm.pl (available electronically) is a sample disassembler that demonstrates how you call the disassembler core function *X86_Disasm* when building disassemblers. Essentially, the disassembler takes your input (in hex format, separated by space), splits them into bytes, and save those bytes in memory. It then calls *X86_Disasm* to disassemble one byte at a time, and prints the disassembled result on the screen. Remember the first item you feed into the disassembler should always be the operand size (16 or 32). The disassembler keeps running until you type "q" to quit. You can run the program on under UNIX or Windows 95/NT.

## Branch-Trace Message Analysis

One of the uses of the disassembler core in this project is to decode the branch-trace messages (BTM) collected from a logic analyzer. The disassembler is called by a program that can parse the bus cycles and extract BTM data by figuring out the target linear addresses and source linear addresses.

Once the code is retrieved, the disassembler can translate the code back into text-like assembly code by calling a low-level memory-read subroutine through *ImageByteRead*. I've tested the disassembler package on both UNIX and Windows platforms. The disassembled code matches the original source code (written in assembly) very well.

## Conclusion

Clearly, there are many improvements you could make to this disassembler. You might want to rewrite the code for Example 2(a) to Example 2(b) if you can make *ImageByteRead* return more than one byte at a time, so that you can reduce the redundancy of the *for* loop. Also, you can add a subroutine to select a test register determined by the *reg* field of the ModR/M byte. Finally, you can add the part for the coprocessor (floating-point unit) instruction set, similar to the way the integer instructions were added in the article.

Searching, extracting, and pattern matching are three things Perl does very well. Perl lets you concentrate on logic flow and algorithm design while it handles the implementation details—that's the beauty of programming in Perl.

## Acknowledgment

I'd like to thank Dr. Ed Ferguson at Texas Instruments for his help in designing the x86 disassembler. Dr. Ferguson originally wrote a disassembler in awk.

**DDJ**

**The Fastest Way To The NT™ DDK**

**DDK WinDK™**

**DDK Class Library**

**Get A Jumpstart On WDM!**

**WinDK is an NT Kernel Level C++ Toolkit**

**WinDK's Features Include:**

- **No DDK Limitations** - WinDK based drivers execute as fast as drivers written using only the NT DDK. No DDK API is restricted.
- **The Samples You REALLY Need** - WinDK comes with samples you can actually use and understand, including:
  - Serial Driver
  - Parallel Port Driver
  - A/D (Analog to Digital) Driver
  - Slave DMA Driver
  - PCI Bus Master Driver
- **Streamlined Driver Creation**: Concentrate on controlling your hardware, not on deciphering the DDK.
- **Develop drivers for multi-processor and RISC platforms.**

**Your Customers Are Going To NT, Shouldn't You?**

**BlueWater Systems INC.**

**800.962.2114**

Tel 206.771.3610 • Fax 206.771.2742
E-Mail: info@bluewatersystems.com
NO RUNTIME ROYALTIES
VISA, MC, AMEX & Approved P.O. accepted

**WinDK Includes:**

- Extensive C++ Class Library
- Real World Usable Samples
- Comprehensive Documentation
- A 300+ Page Manual
- An Easy to Follow Tutorial
- Competent Technical Support

*Why start from scratch?* Use the thousands of lines of debugged code tested on Multi-processor and RISC systems

**FOR HASSLE FREE HARDWARE CONTROL USE**

Version 2.0 **WinRT™**

Windows95™ ← Win32® Binary Compatible → NT™

√ Port I/O √ Memory I/O √ Interrupts
Ask us about Alpha™, PowerPC™ & MIPS versions

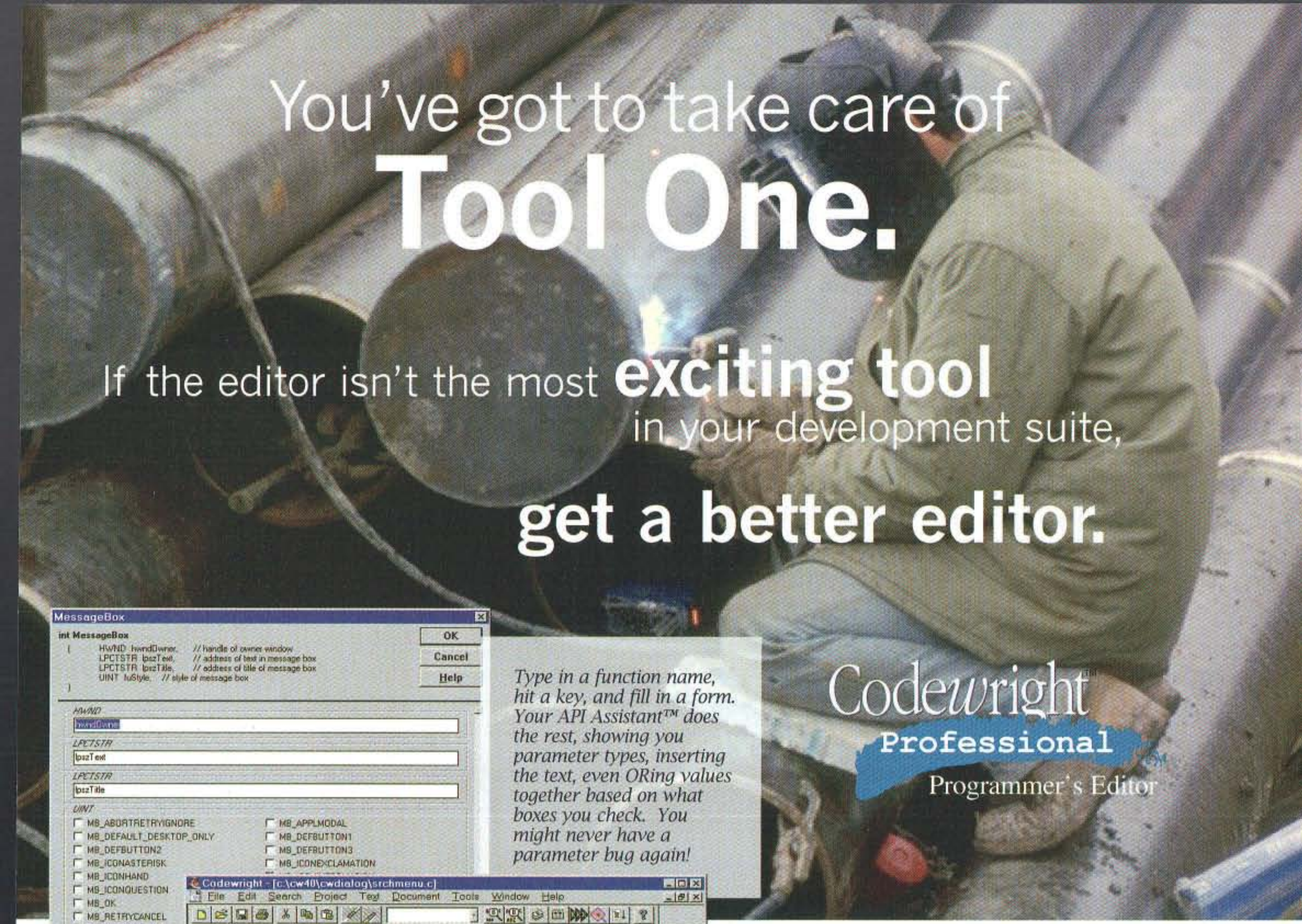
**New Features:**
- Script Language
  - -if - else
  - -while( )
  - -Arrays
- Faster Execution!
  - -Avg. 15% speed increase!
- PCI Support

Also available in OCX Version

Designed for Microsoft® Windows®95

**Please visit our web page or call for more technical information**
**Web Page: http://www.bluewatersystems.com**

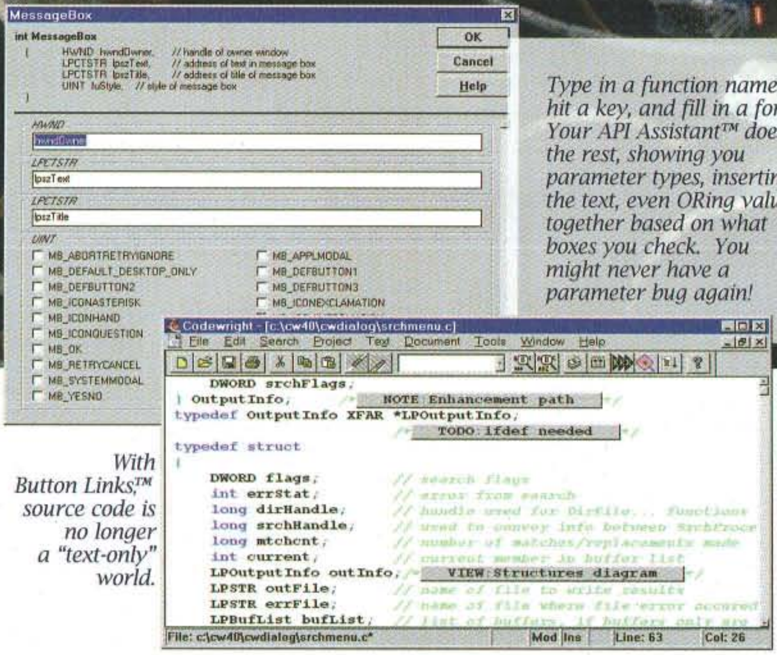
CIRCLE NO. 180 ON READER SERVICE CARD



# You've got to take care of **Tool One.**

If the editor isn't the most **exciting tool** in your development suite,

## get a better editor.



```
MessageBox
int MessageBox
(       HWND hwndOwner,     // handle of owner window
        LPCTSTR lpszText,   // address of text in message box
        LPCTSTR lpszTitle,  // address of title of message box
        UINT fuStyle,   // style of message box
)
HWND       hwndOwner
LPCTSTR    lpszText
LPCTSTR    lpszTitle
UINT
[ ] MB_ABORTRETRYIGNORE          [ ] MB_APPLMODAL
[ ] MB_DEFAULT_DESKTOP_ONLY      [ ] MB_DEFBUTTON1
[ ] MB_DEFBUTTON2                [ ] MB_DEFBUTTON3
[ ] MB_ICONASTERISK              [ ] MB_ICONEXCLAMATION
[ ] MB_ICONHAND
[ ] MB_ICONQUESTION
[ ] MB_OK
[ ] MB_RETRYCANCEL
[ ] MB_SYSTEMMODAL
[ ] MB_YESNO
```

```
Codewright - [c:\cw40\cwdialog\srchmenu.c]
    DWORD srchFlags;
} OutputInfo;        /* NOTE Enhancement path */
typedef OutputInfo XFAR *LPOutputInfo;
                     /* TODO ifdef needed */
typedef struct
{
    DWORD flags;            // search flags
    int errStat;            // error from search
    long dirHandle;         // handle used for DirFile... functions
    long srchHandle;        // used to convey info between SrchProc
    long mtchcnt;           // number of matches/replacements made
    int current;            // current member in buffer list
    LPOutputInfo outInfo; /* VIEW Structures diagram */
    LPSTR outFile;          // name of file to write results
    LPSTR errFile;          // name of file where file error occured
    LPBufList bufList;      // list of buffers, if buffers only are
File: c:\cw40\cwdialog\srchmenu.c*    Mod Ins    Line: 63    Col: 26
```

*Type in a function name, hit a key, and fill in a form. Your API Assistant™ does the rest, showing you parameter types, inserting the text, even ORing values together based on what boxes you check. You might never have a parameter bug again!*

Codewright™ **Professional** Programmer's Editor

*With Button Links,™ source code is no longer a "text-only" world.*

**An editor** is arguably the most essential tool in a developer's toolbox. For many programmers, editing power is the answer to working smarter -- not harder. It does so many things for them that it's their #1 productivity tool.

Productivity? Okay, but when was the last time that editor technology got you breathing heavy? Well, Codewright is packed with innovations that will make you sit up and take notice! Maybe that's why it's the best selling editor for Windows development. That's why it's Tool One.

Now, thanks to Codewright's "Sync" technology, there's no reason to settle for a built-in editor that merely gets the job done. If you use Microsoft's Visual C++ or Borland's Delphi, just integrate with Codewright and you've taken care of Tool One.

While Codewright doesn't *replace* the editor in your environment, Codewright *mirrors* it by loading the same files and reflecting the same edits, regardless of which editor you are using.

To feel the excitement, you'll just have to use Codewright. Call today about a free evaluation.

Call now: 1-888-4-PREMIA
477-3642

http://www.premia.com

Single user license:
Codewright Professional
for Windows, Windows 95, and NT.

**$269**

Volume discounts and
site licenses available.

premia® Premium Quality Software

*Premia Corporation*
*1075 NW Murray Blvd., Suite 268*
*Portland, Oregon 97229 USA*
*Fax: +1-503-641-6001*
*Phone: +1-503-641-6000*
*Email: sales@premia.com*

**CIRCLE NO. 121 ON READER SERVICE CARD**

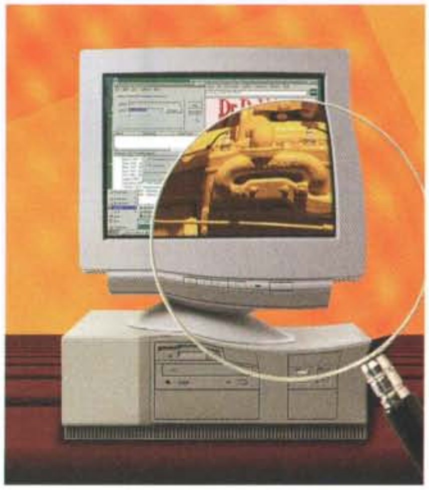
# Examining the Windows NT Filesystem

*A layered organization with filesystem and hardware-device drivers*

Mark Russinovich
and Bryce Cogswell

The Windows NT filesystem is based on the same principles of layering as the Windows 95 filesystem (see our article, "Examining the Windows 95 Layered File System," *DDJ*, December 1995), but its device-driver model makes its implementation significantly different. In Windows 95, for instance, the filesystem consists of 32 distinct layers, each responsible for specific tasks, such as resolving path names and converting from logical to physical offsets. Device drivers, upon loading, indicate to the I/O manager the devices for which they wish to be on the I/O request path for and which layers on the paths they will occupy. In Windows NT, on the other hand, all I/O is device-object oriented and the layering protocol is built by the drivers themselves. Drivers create device objects and link them into chains associated with each physical or network drive. When a request passes down a chain, drivers associated with the device object receiving the request are notified via their associated handler procedures.

*Mark is a consulting associate for Open Systems Resources and can be contacted at mark@meatball.osr.com. Bryce is a researcher at the University of Oregon and can be contacted at cogswell@cs.uoregon.edu.*



In this article, we'll open up the inner workings of the NT filesystem by describing how a filesystem request originates in a user's program and ends up as a disk access. In addition, we'll present an application called "Filemon" that monitors and displays all filesystem activity—paging, opening/closing files, reads/writes, directory searches, and the like. (This mirrors the filesystem API-monitoring capability of the Windows 95 VxD-level call, *IFSMgr_InstallFileSystemApiHook*.) filesystem API monitoring is useful for a variety of purposes, including profiling, virus detection, and security enhancement.

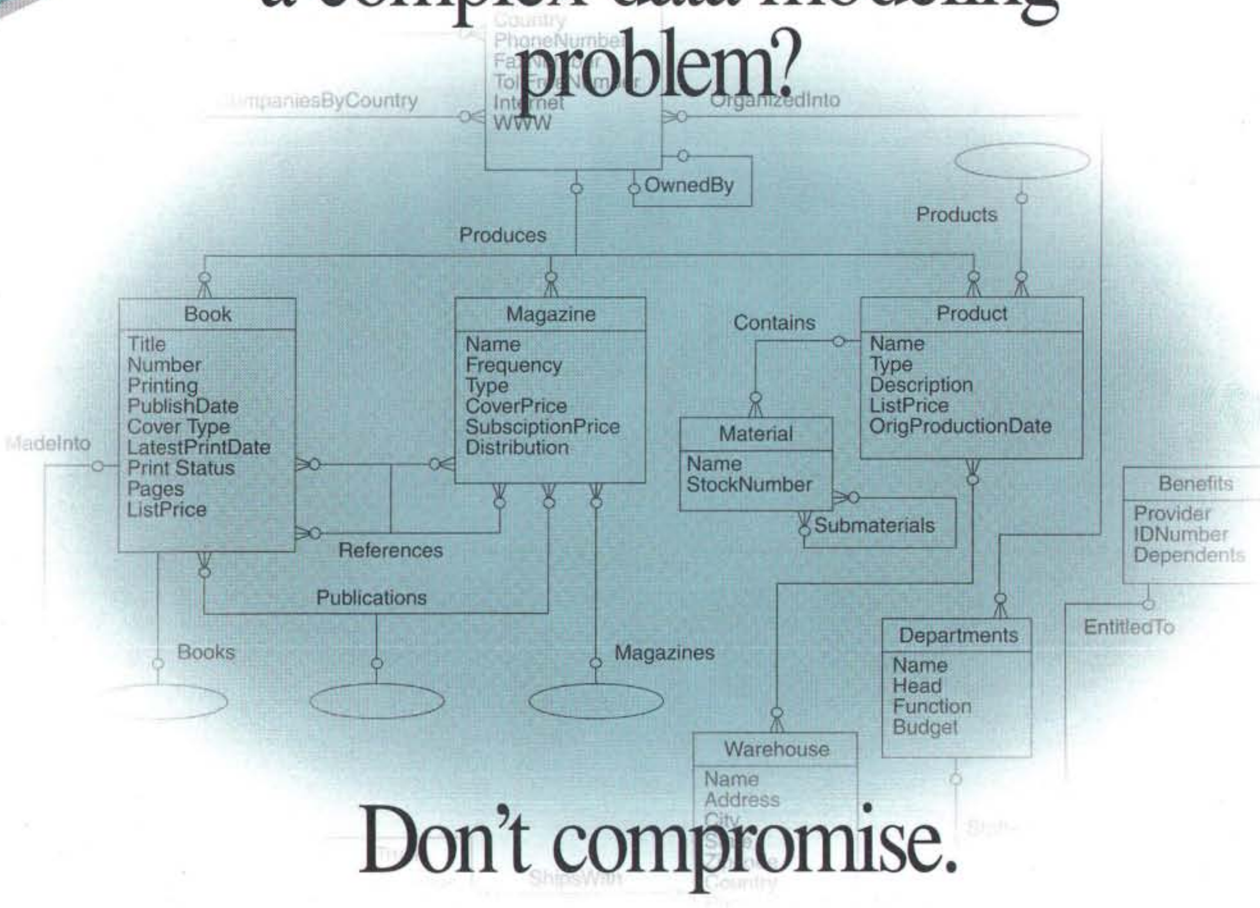
## The Windows NT Filesystem

Any discussion of the NT filesystem requires an introduction to the NT device-driver model. The Windows NT I/O system is based on the concept of object-oriented, packet-driven I/O. This means that I/O requests (IRPs) are created as discrete packets of information and that their destinations are always some type of device object. Device objects are created by device drivers to represent physical, logical, or virtual devices. A device object's driver is responsible for processing IRPs that are targeted at its device object. The I/O system serves as the glue that connects device objects and drivers. Besides providing infrastructure support routines, the I/O system works behind the scenes to route IRPs to target drivers and propagate return values back to the request-initiating driver.

Because the NT I/O model does not make assumptions about how responsibilities will be divided among device drivers, it provides the drivers with the ability to locate objects by name. It also allows device objects to be attached to each other. This enables one to act as a filter for another, processing and taking decisions on IRPs before it sends them on to their destination. To make a device object visible to user-level applications, the driver exports a name to the \DosDevices

Attention Database Developers:

# Do you have a complex data modeling problem?



## Don't compromise.

**O**ther databases require design concessions when modeling complex data. The TITANIUM™ database engine directly supports highly complex relationships, without redundant data, without intersect tables, without joins... without compromise.

Parts explosion problems, for example, and other many-to-many, recursive, and n-ary modeling constructs are handled simply and elegantly by TITANIUM's patent-pending Dynamic Pointer Array (DPA) technology.

## Solve it with

TITANIUM
DATABASE ENGINE

**Optimal Performance for Complex Development Needs**

TITANIUM is a multi-threaded client/server and stand-alone database management system, featuring navigational API's for all major programming languages, SQL/ODBC support, compiled "C" stored procedures, fault-tolerance, transaction processing, and on-line backup — all contained within an extremely small memory footprint. Supported platforms include DOS, Windows (3.x, 95, NT), OS/2, NetWare, and UNIX.

*For a TITANIUM Technical White Paper, call* ***800-445-mdbs*** *or* ***317-463-7200***.

*E-Mail: info@mdbs.com WWW: http://www.mdbs.com*

Successful Applications Are Built With TITANIUM.

mdbs® Micro Data Base Systems, Inc.

**CIRCLE NO. 146 ON READER SERVICE CARD**

*(continued from page 42)*
namespace inside the kernel. (Objects in the kernel namespace can be examined using the WinObj utility provided with the NT 4.0 SDK.) Applications can reference names exported there and have their requests directed to the associated drivers.

The layering supported by NT's driver model introduces three types of drivers, defined by how they interact with other drivers:

- Highest-level drivers create device objects with exported user-space names and receive requests generated directly by user-level applications.
- Lowest-level drivers receive requests only from other device drivers and act as the interface to the lowest level of abstraction a device supports—usually hardware such as a hard disk.
- Intermediate-level drivers act as coordinators between a higher-level driver's requests and lower-level drivers: routing requests, providing transparent error retry, or simply acting as filters.

The generality of NT's approach to device drivers means that a common infrastructure is used to support all types of devices—physical (keyboards, sound cards) and logical (disk partitions). It also means that the I/O system knows little about the filesystem, since this abstraction is provided by the drivers themselves. Consequently, it is difficult to describe the NT filesystem as having an absolute organization; its architecture is not defined by the NT I/O system. However, there is a de facto standard for the general structure. The support provided by the I/O system, as well as the existing base of Microsoft filesystem and hardware drivers, make it much easier to follow design guidelines that are in use than to implement a new architecture.

As Figure 1 illustrates, filesystems currently found on NT generally have only two layers. The first layer consists of filesystem drivers, which define the layout of data in the filesystem; these communicate with the second layer, which consists of the lowest-level hardware drivers—those that read and write information directly on the underlying storage medium. Additional layers can be present, though: Some hardware devices have drivers divided into sublayers where higher levels deal with classes of devices and lower levels address specific types. Furthermore, intermediate-level drivers are often inserted between a filesystem and the lower-level drivers to provide additional functionality such as disk mirroring.

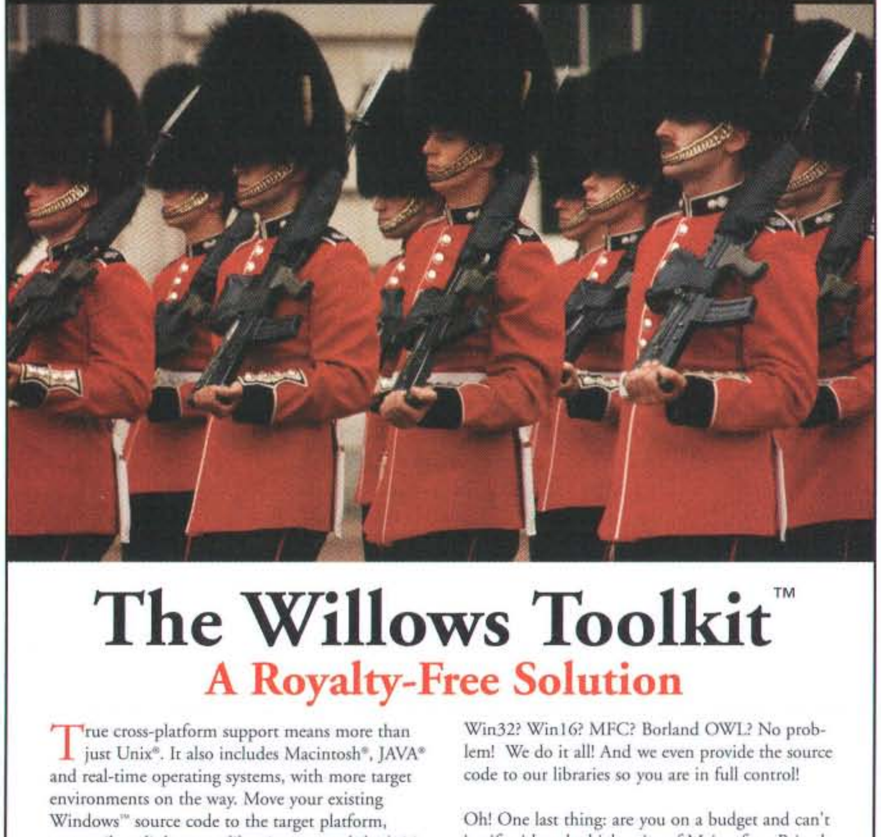
Figure 1 doesn't show how the I/O supervisor's glue connects the drivers, via device objects, nor does it show the kinds of information sent in a request and how that information is packaged. These details are shown in Figure 2, which follows a request through several layers from a Win32 program down to a hard disk and back.

## File and Device Objects

File objects represent files in the same way device objects represent devices. All NT filesystem requests must have a file object as their reference; see Listing One (listings begin on page 85). The first task of the I/O supervisor is to take the user's request; if the request does not reference a previously opened handle, the supervisor creates a file object. A handle to the object is returned to the application, and further operations on the same file can be initiated by referencing the handle. Calls that create a file object, such as the *CreateFile* call, must supply enough information so that the filesystem knows precisely which file is being opened. This information is either a full path name (C:\foo\bar.txt), or an implied current directory (C:\foo) and a relative path name (bar.txt).

The I/O supervisor determines the request's target logical drive (in this case, C:\) by extracting it from the full path name. It then determines the physical partition on which the drive resides. This is done by looking up the drive letter in \DosDevices, the kernel's internal exported-name directory. Logical drives are normally represented in this internal table with symbolic

Royalties! No Way....



The Willows Toolkit™

A Royalty-Free Solution

True cross-platform support means more than just Unix®. It also includes Macintosh®, JAVA® and real-time operating systems, with more target environments on the way. Move your existing Windows™ source code to the target platform, recompile, relink to our libraries, test and that's it! You now offer native support and native performance of your applications on many more operating systems with little incremental effort!

Win32? Win16? MFC? Borland OWL? No problem! We do it all! And we even provide the source code to our libraries so you are in full control!

Oh! One last thing: are you on a budget and can't justify either the high price of Mainsoft or Bristol toolkits or the $75 to $200 royalty they levy against each copy of the application you develop and ship to your end-user? Then call us today!

Download your free evaluation copy today!
www.willows.com

WILLOWS SOFTWARE

**Willows Software, Inc.** 12950 Saratoga Avenue, Suite A, Saratoga, CA 95070
2727 Philmont Avenue, Huntingdon Valley, PA 19006
(800) 814-6370 or (408) 777-1820 Fax: (408) 777-1827 info@willows.com

CIRCLE NO. 274 ON READER SERVICE CARD

links to another name that identifies the device object representing the drive. A reference to this device object is stored in the file object so that future requests can skip this look-up step.

For example, a system where C:\ is the first partition on the first hard disk might have an entry "\DosDevices\C:\" that is pointing to a symbolic link with the value "\device\harddisk0\partition1." Once it has found the device object for the drive, it determines which filesystem device object is associated with it by looking at the partition's volume parameter block— a data structure in the device object that links a partition to a filesystem.

With the filesystem device object in hand, the I/O supervisor creates an IRP for the request. An IRP is simply a data structure (see Listing Two) that carries information about a specific filesystem request. This includes the request type (open, read, write, query, and so on) and associated file object (the handle of the file to be operated upon), as well as storage for intermediate driver-specific information, buffer pointers, and status flags. The I/O supervisor is responsible for sending the newly created IRP to the top-level device driver on the chain associated with the target drive, typically the filesystem driver.

## Drivers

Each driver in the chain has multiple dispatch entry points, dedicated to processing specific types of filesystem requests. The set of entry points is defined by the *MajorFunction* table in the driver object. The I/O supervisor invokes the driver via the *IoCallDriver* call, passing the target device and IRP as arguments, and indexing into the driver's table of registered request-type handlers. There it finds the procedure in the device driver that processes the particular request type.

At this point, the filesystem driver takes over. Its actions depend largely on the type of filesystem it serves and the type of request it has been sent. For accesses to files on a local FAT, HPFS, or NTFS drive, the filesystem driver is FASTFAT, HPFS, or NTFS, respectively. For accesses to a CD-ROM, the driver is usually CDFS, while for network drives, the driver is usually NWRDR, the network redirector filesystem. Looking again at Figure 2 (where the access is to a FAT partition), we can see that FASTFAT determines which logical sectors on the drive must be accessed to complete the request. It then creates one or more new IRPs that it sends to the device object representing the partition, in this case the one named "\device\harddisk0\partition1." The I/O supervisor routes the IRPs to the correct handler procedure in the device driver that owns the partition's device object. The example path ends up in the ATDISK device driver, which is the standard IDE-type hard-disk driver. ATDISK interacts with the hard disk, and upon completion of the disk action, sends a message back to FASTFAT, indicating that it has finished servicing the request, passing a status value back in the IRP. The return route continues back up the driver chain, and, as indicated by the red lines in Figure 2, does not go through the device-object path.

The path and procedure followed by a CD-ROM, floppy, or other local request is similar to Figure 2. If the target drive is a network drive, however, the filesystem driver does not perform the task of translating the request to logical sector accesses. Instead it serves merely as a proxy for a filesystem driver, such as FASTFAT or NTFS, residing on the remote node. It forwards the request to the remote node using a device driver for the network card on the local machine and then presents the results to the user application when they are received from the network.

## Variations

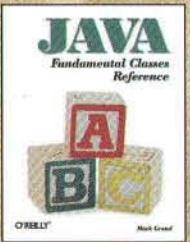
The process whereby the I/O supervisor creates an IRP to send to filesystem drivers allows for great flexibility in the way a driver can respond to a request. It can reply immediately with a negative result (for example, if it determines the

WEB MASTERY

LIKE AN EXPERT PEERING OVER YOUR SHOULDER, O'REILLY BOOKS GIVE YOU THE REAL-WORLD ADVICE YOU'RE LOOKING FOR.

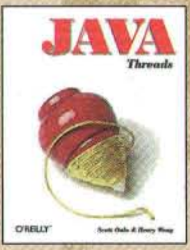
We write books that help you master Internet publishing and programming topics, going above and beyond mere mechanics. Our books explore subjects in-depth and are written and reviewed by experts. Next time you encounter a problem, reach for the solution that doesn't require bits or bytes or RAM. Reach for an O'Reilly book.

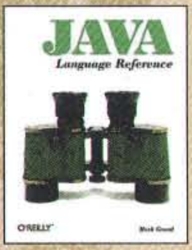
Java Fundamental Classes Reference
*By Mark Grand*
*1st Edition Winter 1997*
*500 pages (est.)*
*ISBN 1-56592-241-7*
*$34.95 (est.)*



Java AWT Reference
*By John Zukowski*
*1st Edition Winter 1997*
*400 pages (est.)*
*ISBN 1-56592-240-9*
*$39.95 (est.)*

Java Threads
*By Scott Oaks & Henry Wong*
*1st Edition Winter 1997*
*275 pages (est.)*
*ISBN 1-56592-216-6*
*$29.95 (est.)*





Java Language Reference
*By Mark Grand*
*1st Edition Winter 1997*
*450 pages (est.)*
*ISBN 1-56592-204-2*
*$32.95 (est.)*

O'REILLY™

101 Morris Street, Sebastopol, CA 95472 • 707-829-0515 • Orders/inquires: 800-998-9938
Weekdays 6AM-5PM PST • Fax: 707-829-0104 • Email a request for our catalog: catalog@online.ora.com
Browse our catalog on the Web: http://www.ora.com/
O'Reilly books are also available at your local bookstore. • Prices do not include shipping and handling
Please mention Code ADDJ when ordering.

CIRCLE NO. 168 ON READER SERVICE CARD

file or file handle does not exist); it can take the IRP and pass it to low-level drivers; or it can create several new IRPs to send to lower drivers.

The NT I/O model makes special allowance for the performance demands of filesystem drivers in an optimization that can sometimes be used to avoid the creation of an IRP. This backdoor to the filesystem is known as the "fast I/O path." A filesystem device driver can specify in its driver-object data structure a table of routines that can be called directly to handle simple requests. The I/O supervisor uses one of these shortcuts when it is likely that a filesystem access will be served from the disk cache. If the filesystem driver cannot satisfy the request without calling lower-level drivers, it fails the fast I/O call and the I/O supervisor proceeds with normal IRP creation. This fast path saves the memory allocation and setup required to perform an IRP-based request.

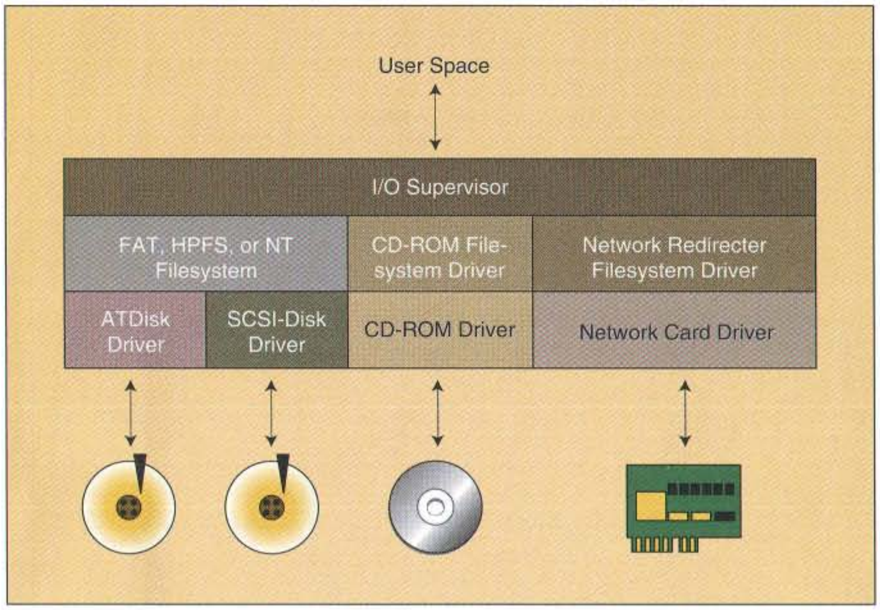

User Space
I/O Supervisor
FAT, HPFS, or NT Filesystem
CD-ROM File-system Driver
Network Redirecter Filesystem Driver
ATDisk Driver
SCSI-Disk Driver
CD-ROM Driver
Network Card Driver

***Figure 1:*** *Typical Windows NT filesystem organization.*

## The Filemon Application

To demonstrate this device-driver and device-object model, we have implemented an application called "Filemon" that layers above filesystem drivers by creating hook-device objects and attaching them to logical-drive device objects. Filemon's device driver sees every request directed to any of the system's logical drives.

The Filemon application, which runs under NT 3.51 and 4.0, consists of two separate components — a Win32 GUI called "filemon.exe" (see Figure 3), and an NT device driver called "filemon.sys." Filemon.sys is a dynamically loaded device driver; no installation procedure is necessary since filemon.exe loads it. During initialization, Filemon determines the types of logical drives present and lets users select the set of drives to be monitored, including hard disks, CD-ROMs, floppies, network drives, and RAM drives.

During execution, Filemon dumps information about all requests directed at the set of monitored drives. This information includes number of requests, request type, path name of the associated file, result status, and any request-specific information such as file offsets and lengths.

## Information-Handling Issues

A frequently seen request type is IRP_MJ_CLEANUP, and its purpose is a little less obvious than most of the requests. Each file object has both an open count and a reference count associated with it. The open count is the number of open user-level handles that currently refer to the object. The reference count indicates the number of references, made by open handles as well as by other objects in the kernel, to a file object. The open count is always less than or equal to the reference count. When the open count drops to zero, the I/O supervisor sends the filesystem a cleanup request telling it that any outstanding IRPs it may have for the file object need not be completed since no user is waiting for a result. When the reference count drops to zero, the file object can be closed, and the filesystem will receive an IRP_MJ_CLOSE request.

• Windows NT • Solaris • Windows 95 • PC-DOS • Theos •

Coherent Unix • Lynx OS • Free BSD • NetWare • CTOS

OS/2 • NextStep • UnixWare • XENIX • THEOS • DOS

**Run more than one OS on your PC!**

SYSTEM COMMANDER

*System Commander* makes it safe and easy to add as many OSes to your PC as you want!

**Easy, Automatic Installation**

- Prepares your PC for new OSes in minutes.
- Simply select the OS you want and System Commander does the rest.
- To use another OS just reboot and select another selection.

"Highly recommended"
John C. Dvorak - PC Magazine

**Money Saving Flexibility**

- Use any Intel compatible OS or combination of OSes, in English or any other language on one PC.

***System Commander*** is only $99.95 with a 60 day unconditional money back guarantee. **Free overnight shipping** when you mention this ad.* Call now and have it tommorrow.

**V Communications, Inc.**
**800•648•8266** DD
**www.v-com.com**

*When ordered before noon PST. No Saturday delivery. Standard shipping outside US. CA res.add $7.25 sales tax. Offer subject to change without notice. All logos and product names are trademarks of their respective companies. VISA/MC/Amex ©1996

• Windows v3.x • DOS/V • Warp • Windows 95J • QNX •

CIRCLE NO. 155 ON READER SERVICE CARD

# DEAR DR. MORTY

## *Today's Topic*

How to choose a software configuration management system

**DEAR DR. MORTY: WHAT EXACTLY IS SCM AND HOW DOES IT APPLY TO MY JOB?**

A. Software configuration management (SCM) is the process of tracking and managing software development projects of any kind. A quality SCM solution should provide such functionality as version control, workflow features, and advanced security. It enables development teams to deliver mission critical software or even create new intranet-based applications in a controlled, project oriented manner.

**Q. HOW WILL SCM BENEFIT MY R&D TEAM?**

A. Your entire team—from project leaders to developers and testers—will benefit from a controlled development process. Properly managed source code prevents bottle-necks in development, testing and QA, eliminates code collision and increases productivity. It enables you to easily retrieve earlier versions of code so your team can respond quickly to market demands. The bottom line is that properly managed code becomes shipping code *faster.*

**Q. WHAT SHOULD I LOOK FOR IN AN SCM TOOL?**

A. Control and productivity is the key. Consider basic functionality such as easy check-in and check-out; project check-pointing to quickly re-create past releases, and re-use of code across projects—a real time saver! Keep in mind the benefits provided by individual work environments that allow team members to work independently of other changes. A strong security feature is a *must* to ensure protection for your valuable source code.

**Q. HOW IMPORTANT IS INTEGRATION INTO VARIOUS DEVELOPMENT ENVIRONMENTS?**

A. Very important, but don't be fooled. All hooks are not created equal. Look for a tool that will let your team work in their preferred development environment, with seamless integration back into the SCM system.

**Q. I WASTE A LOT OF TIME FIGURING OUT CHANGES BETWEEN FILES—IS THERE AN EASY WAY TO MERGE CODE?**

A. Good question. Merging code is easy, especially if your SCM solution provides a "Visual Differencing" and a "Visual Merge" function. Most products allow you to see differences in code—additions, deletions, changes—by color coding, but the ability to merge these changes as you see the differences will really bump up your productivity.

**Q. HOW EASY IS IT TO GET MY TEAM STARTED?**

A. Choosing the right SCM tool is a decision you shouldn't take lightly. Consider things like the number of platforms you'll be running on, which policies your company wants to implement, the importance of security, and the location of your existing files. Installation of the software should be simple (depending on what tool you select). Adding existing files to the new system should be as easy as a few clicks of a mouse.

**Q. IS SCM EXPENSIVE?**

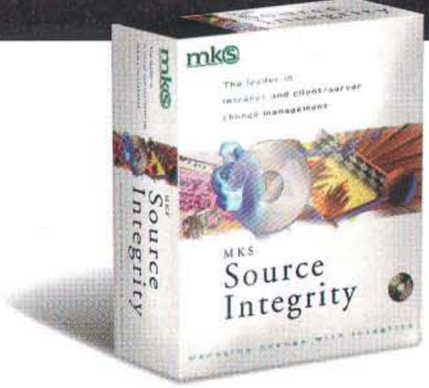
A. When you select an SCM system, cost is always a factor. Look for a comprehensive solution that meets all your criteria, right out of the box. You don't want to drain your budget to keep it running after installation, or continually add costly pieces of functionality that weren't included in the original price.

Until next time, this has been Dr. Morty on how to choose a software configuration management system.

**And remember, change is good.**

You can write to Dr. Morty at: morty@mks.com for your specialized SCM requirements.

## The leader in software configuration management from Client/Server to Intranet.



With MKS Source Integrity, your development team will benefit from such powerful features as an advanced security and administration module (SAM), variant sandboxes, freeze and thaw, file locking and more. The ability to check in and out related files together, automatically reload altered working files, and manage any file type will streamline the development process even more.

### Increase your productivity

Time saving-features like Visual Differencing and Visual Merge, project labeling and event triggers allow for smoother work flow and increased productivity. Benefit from the ability to reflect multiple promotion models, customizable reporting, and integrated MAKE build and release support as your team's needs grow.

### Cross-platform support

A true cross-platform solution, MKS Source Integrity supports Windows, OS/2 and ten flavors of UNIX. In fact, MKS Source Integrity provides integrated version control to leading Web servers including Netscape Enterprise Server and Microsoft Internet Information Server. MKS Source Integrity is a perfect tool for Web development.

### True integration

MKS Source Integrity provides built-in functionality from your native development environment including PowerBuilder, Microsoft Visual Basic, Microsoft Visual C++, Borland C++, Borland Delphi, Watcom C++ and more.

### Easy to use

With MKS Source Integrity you can be up and running in literally minutes. You can put your files under version control with point and click simplicity. MKS Source Integrity makes your team instantly productive.

### Iron-clad

You can count on MKS's expert technical support, Professional Services and Training, award-winning products, and unconditional money-back guarantee. That's why leading companies such as Cognos, Intel, 3M and CASE trust their source code to MKS.

### Try it free

**Call today: 1-800-265-2797 for more information, or point your browser to www.mks.com for your free demo.**

International customers may call
MKS France +33 1 3082 2762,
MKS Germany +49 711 16714 0,
MKS Nordic +45 3325 6555 or
MKS UK +44 171 624 0100.

mks MORTICE KERN SYSTEMS INC.

SOFTWARE Development PRODUCTIVITY AWARD 1995

MKS and MKS Source Integrity are registered trademarks of Mortice Kern Systems Inc. All other trademarks acknowledged.

CIRCLE NO. 94 ON READER SERVICE CARD

# Embedding Microsoft® Windows® Is Easier Than You Think.

**The One-Stop Source**

Imagine running Microsoft MS-DOS 6.22, Windows 3.1, and a Visual Basic application all on a 2-Megabyte Flash disk!

AnnaSoft has all the tools you need to tailor MS-DOS and Windows – and even your BIOS – to exactly match your system's objectives for size, capability and performance.

**Licensing**

And, as an OEM you can license any version of MS-DOS, Windows (including 95 and NT) and other Microsoft system support products directly from AnnaSoft!

**Support**

To wrap it up you'll get on-line technical support from designers who have been there. Call us toll free for all the facts.

**1-800-690-3870**

AnnaSoft
AN ANNABOOKS AFFILIATE

11838 Bernardo Plaza Court
San Diego, CA 92128-2414
(619) 674-6155 · Fax: (619) 673-1432
e-mail: four11@annasoft.com
On line at http://www.annasoft.com/ddj

Microsoft, MS-DOS and Windows are registered trademarks of Microsoft Corporation

*(continued from page 46)*

On an unrelated note, Filemon's device driver dynamically allocates storage in which to save filesystem request records. Although rare, it is possible for a system to generate requests fast enough that buffering all requests for the GUI would require too much memory. If that happens, some records may be dropped intentionally, reflected as a gap in the sequence of records displayed by the GUI.

There are two situations where the pathname displayed is a drive letter followed by the text "DIRECT I/O." One is when the disk cache is flushing data that has no associated name. For example, in the FAT filesystem, the file-allocation table itself has no name. However, it is still desirable to cache file-allocation tables for quick access. The FASTFAT driver therefore has to cache the data as a direct I/O file object. Later, when the cache wants to flush out to disk, FASTFAT will receive a request with byte offsets relative to the start of a logical drive as a reference, rather than relative to the start of a file.

Direct I/O is also used by administrative software such as FORMAT or CHKDSK. These utilities write directly to disk sectors, bypassing the structure imposed by filesystem drivers. If you format a floppy disk while running Filemon, for instance, you will see FORMAT accessing the floppy disk via direct I/O.

An interesting result of the asynchronous behavior of many filesystem requests is the possibility of outstanding IRPs at GUI shutdown. When this is the case, Filemon's device driver cannot unload, because when an outstanding IRP completes, the I/O supervisor will attempt to call the I/O completion routine that Filemon has registered, regardless of whether it is in memory. (You can imagine the outcome if the device driver has been unloaded.) Under these circumstances, the GUI exits, but the driver remains memory resident to handle these last IRP completions. If another instance of the GUI is started, it connects with the already loaded driver image; at exits of these subsequent GUI instances, a check is made again to see if the driver can unload cleanly.

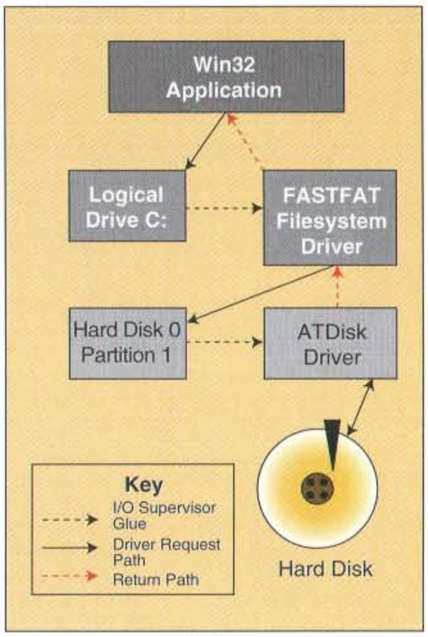
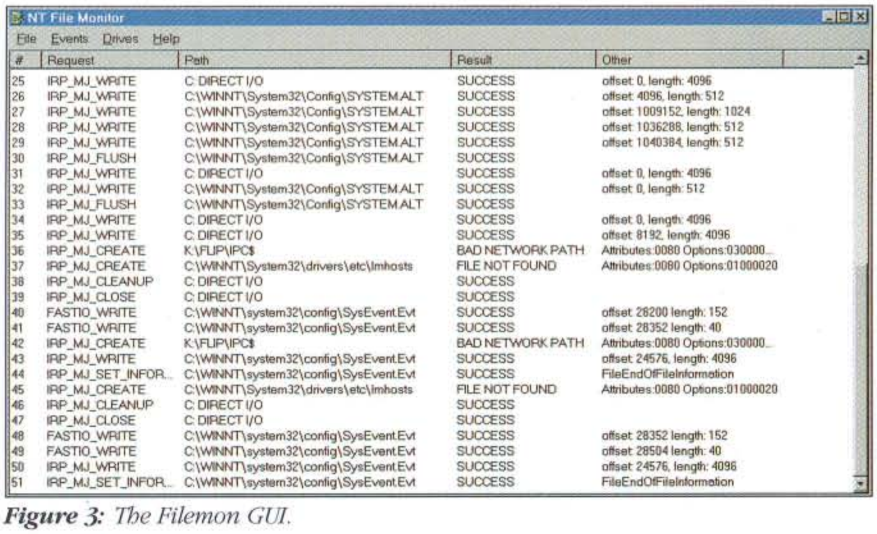


**Figure 2:** *Example filesystem control flow.*

## How Filemon Works

When the GUI starts, it initiates a load of the Filemon device driver (see the INSTDRV example in the NT DDK under SRC\GENERAL\INSTDRV for an example of dynamically loading a device driver). During initialization, the device driver creates a device object with the kernel object name "\DosDevices\FILEMON," so that the GUI can communicate with it via *CreateFile* and *DeviceIoControl* calls. The GUI then requests the driver to hook the set of drives desired by the user.



| # | Request | Path | Result | Other |
|---|---|---|---|---|
| 25 | IRP_MJ_WRITE | C: DIRECT I/O | SUCCESS | offset: 0, length: 4096 |
| 26 | IRP_MJ_WRITE | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | offset: 4096, length: 512 |
| 27 | IRP_MJ_WRITE | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | offset: 1009152, length: 1024 |
| 28 | IRP_MJ_WRITE | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | offset: 1036288, length: 512 |
| 29 | IRP_MJ_WRITE | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | offset: 1040384, length: 512 |
| 30 | IRP_MJ_FLUSH | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | |
| 31 | IRP_MJ_WRITE | C: DIRECT I/O | SUCCESS | offset: 0, length: 4096 |
| 32 | IRP_MJ_WRITE | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | offset: 0, length: 512 |
| 33 | IRP_MJ_FLUSH | C:\WINNT\System32\Config\SYSTEM.ALT | SUCCESS | |
| 34 | IRP_MJ_WRITE | C: DIRECT I/O | SUCCESS | offset: 0, length: 4096 |
| 35 | IRP_MJ_WRITE | C: DIRECT I/O | SUCCESS | offset: 8192, length: 4096 |
| 36 | IRP_MJ_CREATE | K:\FLIP\IPC$ | BAD NETWORK PATH | Attributes:0080 Options:030000... |
| 37 | IRP_MJ_CREATE | C:\WINNT\System32\drivers\etc\lmhosts | FILE NOT FOUND | Attributes:0080 Options:01000020 |
| 38 | IRP_MJ_CLEANUP | C: DIRECT I/O | SUCCESS | |
| 39 | IRP_MJ_CLOSE | C: DIRECT I/O | SUCCESS | |
| 40 | FASTIO_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 28200 length: 152 |
| 41 | FASTIO_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 28352 length: 40 |
| 42 | IRP_MJ_CREATE | K:\FLIP\IPC$ | BAD NETWORK PATH | Attributes:0080 Options:030000... |
| 43 | IRP_MJ_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 24576, length: 4096 |
| 44 | IRP_MJ_SET_INFOR... | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | FileEndOfFileInformation |
| 45 | IRP_MJ_CREATE | C:\WINNT\System32\drivers\etc\lmhosts | FILE NOT FOUND | Attributes:0080 Options:01000020 |
| 46 | IRP_MJ_CLEANUP | C: DIRECT I/O | SUCCESS | |
| 47 | IRP_MJ_CLOSE | C: DIRECT I/O | SUCCESS | |
| 48 | FASTIO_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 28352 length: 152 |
| 49 | FASTIO_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 28504 length: 40 |
| 50 | IRP_MJ_WRITE | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | offset: 24576, length: 4096 |
| 51 | IRP_MJ_SET_INFOR... | C:\WINNT\system32\config\SysEvent.Evt | SUCCESS | FileEndOfFileInformation |

**Figure 3:** *The Filemon GUI.*

To hook a drive, the device driver must first determine the device object that represents it. It uses the same technique that the kernel uses to determine the logical-drive device object referenced by an access to a previously opened file object (see Listing Three). The device driver calls the kernel's internal open-file call, *ZwCreateFile*, on the root directory of the target drive (C:\, for example). This returns the necessary file-object handle. Next, it maps the handle to an actual file-object pointer, using *ObReferenceObjectByHandle*. Finally, the device object associated with the file object is retrieved with *IoGetRelatedDeviceObject*.

Once a logical drive's device object has been located, the driver creates a hook-device object and attaches it to the drive's object via *IoAttachDeviceByPointer*. Thereafter, the hook object receives all requests directed to the logical drive's top-level driver. Except for dealing with the fast I/O path, the rest of the device driver's chores are straightforward. It simply registers handler routines for requests directed to filesystem drivers and pulls information about requests out of the IRPs it sees on their way to the filesystem. An I/O completion routine is registered for each request so that the driver can see the associated return status.

Handling the fast I/O path is tricky because it is not documented in the NT DDK. A filter driver such as Filemon cannot ignore this control path because the I/O supervisor assumes that filesystem drivers have at least *some* fast I/O calls implemented; ignoring them leads to a quick system crash. Fortunately, the fast I/O call prototypes and table definition are included in the DDK's main include file, NTDDK.H, so adding fast I/O routines to a filter driver is straightforward—once the problem is understood.

Filemon copies ASCII text describing each request to a buffer that it transfers to the GUI periodically. Although only the most significant aspects of each transaction are recorded, the application is easily extended to display any data contained in the IRP. Unfortunately, much of the information packaged with requests is not documented. For many purposes, including enhanced security, profiling, and virus detection, the data collected by Filemon is sufficient. However, until Microsoft produces its much-awaited filesystem documentation, you must be prepared to put on your spelunking gear and get dirty with a kernel-mode debugger if you want a more detailed understanding.

**DDJ**

**(Listings begin on page 85.)**

**High Performance Imaging Development Tools**

**RasterMaster™ 6.0** - powerful, multi-platform, raster imaging development tools and technology. We specialize in **speed, capability, features and reliability.** Specializing in both document imaging and color imaging ***that works as advertised!***

**Familiar API and well-proven technology** in use world-wide by companies such as British Petroleum, Chase Manhattan, Eastman Kodak, Ford Motor, Hewlett-Packard, Philips Consumer, Pharmacia/Upjohn, Polaroid, Xerox and many more.

**40+ Raster Formats**: All TIFF, JPEG, Group 4, Group 3, MO:DCA IOCA, CALS, PNG, PCX, BMP, GIF, Photo CD, IBM ABIC...

**Full Set of Functions**

| | |
|---|---|
| Format Reading, Saving, Compression | Internet GIF support* |
| Display, rotate, zoom, pan, scroll... | Image processing |
| Deskew* | Transparent color* |
| Despeckle* | Improved Twain scanning* |
| MO:DCA IOCA writing* | Color reduction |
| Anti aliased display | Color promotion |
| Auto aspect ratio* | ...and much more |
| Low level control | **- new in version 6.0* |

**Platforms**: Win 95, Win NT, Win 3.1, OS/2, Macintosh and UNIX

**Environments:** DLL's, VBX's, OCX's, Delphi, Visual Basic, Power Builder, Visual C and more

**Call now or see our Website for free, no risk evaluations!**

**SNOWBOUND SOFTWARE...**

Website: http://www.snowbnd.com
Email: salesdj@snowbnd.com
PO Box 520 Newton, MA 02159
Tel: 617 630-9495 Fax: 617 630-0210

**CIRCLE NO. 309 ON READER SERVICE CARD**

# Robots and Finite-State Machines

*Walking the walk with mobile robots*

Everett F. Carter, Jr.

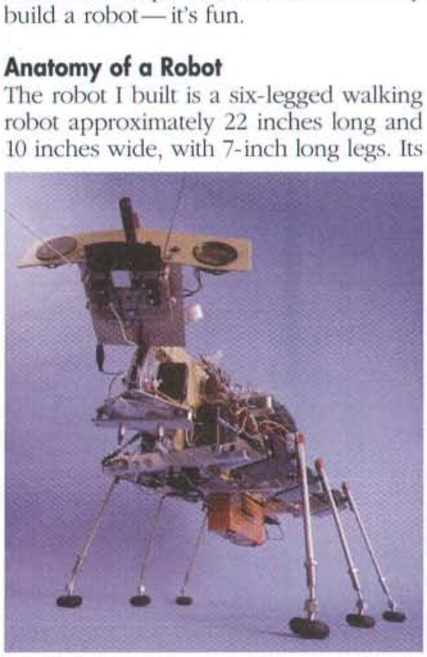
Designing and building autonomous robots presents a host of technical challenges. On the hardware side, for instance, you have to contend with computational and interface electronics, torques, pulleys, gears, and levers. On the software side, the challenges involve real-time data acquisition, real-time process control, artificial intelligence, and other issues familiar to embedded-systems designers. In this article, I'll describe the high-level processing I implemented in a robot designed for researching robot-control systems. (The source code implementing this processing is available electronically; see "Availability," page 3.)

Since the purpose of this robot was to examine the control problems at a relatively high level, you may wonder why I went to the trouble of building a physical robot instead of just developing the necessary control techniques through simulation. The main reason for building the device is that it is extremely difficult to honestly simulate the effects that occur in a real, uncontrolled environment: Infrared sensors can give false positives or negatives, the sonar system can see false echoes (or the sound gets absorbed instead of reflected by an object), motors can stall, power supplies sag, and walking robots can tumble when the center of gravity shifts. All these effects can be a challenge to realistically simulate. Consequently, simulations are necessarily idealistic, and so, to quote R.A. Brooks and M.J. Mataric, "Simulations are doomed to succeed" (Real Robots, Real Learning Problems," *Robot Learning*, edited by J.H. Connell and S. Mahadevan, Kluwer Academic Publishing, 1993). Of course there is an even more important reason to actually build a robot—it's fun.

*Everett is president and senior software developer for Taygeta Scientific and president of the Forth Interest Group. He can be contacted at skip@taygeta.com or http://www.taygeta.com/.*

## Anatomy of a Robot

The robot I built is a six-legged walking robot approximately 22 inches long and 10 inches wide, with 7-inch long legs. Its sensors consist of a pair each of ranging ultrasonic sonars, active infrared (IR) light detectors for proximity detection, and contact sensor whiskers about 8 inches long. The sonar works in the range of about 9 inches to 30 feet, and is used for long-distance object detection and mapping. The IR system is tuned to detect objects from about 14 inches away. The whiskers detect actual contact of the robot with an object.



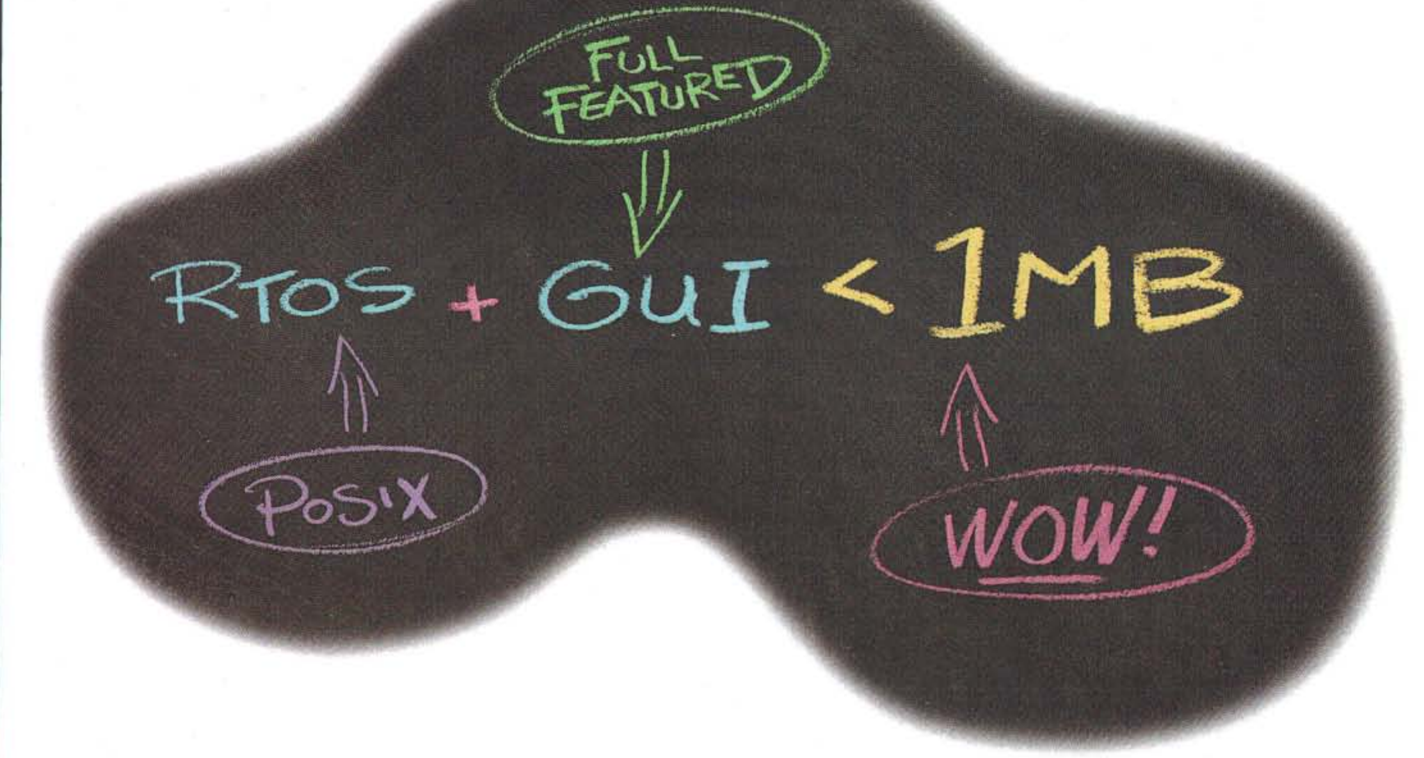
All sensors are mounted on a sensor head that can move +/− 45 degrees to either side, as well as up and down. Putting the sensors on the head eliminates the need for expensive rings of sensors that are common on autonomous robots, but adds the complication of having to know where the head is pointing in order to provide a proper context for the sensor data.

In theory, the sonar system can pick out an obstacle before the IR notices it and both of these should report it before the contact sensor detects it. This redundancy improves the reliability of the robot since each of these object-detection mechanisms uses a different physical principle, making it less likely that they would all fail to see something.

Building a walking robot is considerably more challenging than I originally anticipated. After several frustrating weeks, I learned the techniques required to get a statically stable robot (as opposed to a dynamically stable robot which needs to actively balance itself) to successfully walk. S.M. Song and K.J. Waldron's *Machines that Walk: The Adaptive Suspension Vehicle* (MIT Press, 1989) is an excellent source of information on motion control for this kind of robot. After experimentation, I developed a library of possible gaits for the robot. Each gait consists of a sequence of (what I call) cycles for each of the six legs in a particular order. A cycle consists of:

- Raising the leg.
- Swinging it forward or backward a certain amount.
- Lowering the leg back to the ground.
- Pushing or pulling the leg back to the centered position again.

By combining variations in forward and backward cycles, and in the order of the sequence of legs being moved, the robot can be made to walk forward, backward, turn in either direction, and handle variations in the slope or roughness of the terrain. Song and Waldron describe some very elaborate gaits for doing things such as negotiating steps, crossing trenches, and moving over fences.



## Do more with less.

Most operating systems leave little room for important things. Like your application.

With the QNX realtime OS you'll pack more functionality into less memory. Achieve more performance with less-expensive CPUs. And deliver better solutions at a lower price.

Better yet, QNX supports more PC hardware than any other realtime OS. Whether it's PC/104 or PCMCIA, embedded X86 or the Pentium® Pro, QNX lets you use it right out of the box. Get to run time in less time!

**More scalable than ever!** From low-end to high-end, QNX offers you the ultimate in scalability. Even your deeply embedded systems can boast a scalable POSIX RTOS, thanks to our new, exceptionally small Neutrino™ microkernel.

And if you need to add the capabilities of a high-end GUI to your low-end system, you can. Our award-winning Photon microGUI™ gives you a phenomenal front end, with enough memory left over for important things . . . like your application!

**www.qnx.com**

Call **1 800 676-0566 (ext. 1328)**
or email **info@qnx.com**

QNX®
***The Leading Realtime OS for PCs***

QNX Software Systems Ltd., 175 Terence Matthews Crescent, Kanata, Ontario, Canada K2M 1W8 Voice: 613 591-0931 Fax: 613 591-3579
**Europe:** 49 Dove Park, Chorleywood, Hertfordshire. Voice: (44)(0)1923 284800 Fax: (44)(0)1923 285868 Email: QNXeurope@qnx.com

CIRCLE NO. 134 ON READER SERVICE CARD

*(continued from page 50)*

There are two major categories for walking-machine gaits: periodic and nonperiodic. Nonperiodic gaits are useful in complicated terrain and require foot contact and pressure feedback to be useful. Periodic gates always repeat the sequence of cycles and are easier to implement.

The gaits I generally use with the robot are backward-wave gaits and equal-phase gaits. Wave gaits have the best stability margin and are commonly used by animals. In backward-wave gaits, the phase of the leg placement sequence on each side runs from the front leg to the rear (there is also a forward-wave gait that runs the other way). Equal-phase gaits are not common in nature, but for robots they can be very useful. These gaits distribute the placing and lifting of the legs evenly over the complete motion cycle, which means that the power demand to drive the legs is relatively even. As a result, equal-phase gaits are very popular in walking robots that are hydraulically or pneumatically driven.

Two of the gaits in the library that I use for walking forward are the backward-wave gait sequence (1, 2, 3, 4, 5, 6) and the equal-phase gait sequence (1, 4, 5, 2, 3, 6). By convention, walking-machine legs are numbered from front to back, with odd numbered legs on the left and even numbered legs on the right. These particular sequences do a complete cycle of one leg before starting the cycle for the next leg. Some gaits have overlapping timing of the cycles. In these, the robot can move more quickly, but is less stable.

The robot has a hierarchical design. At the very low levels, I used dedicated microcontrollers to control functions like generating the real-time PCM signal to control the leg servo motors (this is done with an MC146805K1 processor for each leg) and modulating the IR and debouncing the contact switches (this is done with a Basic Stamp PIC controller).

High-level decisions and evaluations of the robot's environment required a microprocessor. The high-level system I used is a single-board MC68332-based system from New Micros Inc. In this article, I'll primarily focus on the software that resides on this system.

Since the robot is designed to be autonomous, there is no special user interface. You communicate with the master CPU via its built-in RS-232 interface and interact with its on-board Forth interpreter/compiler. The Forth system acts as both an operating system and programming language for the robot. Forth is nice for developing robot-control software because it can be incrementally compiled and tested in small parts. For example, to determine how high to step while walking, I mounted the robot on its test stand (a camera tripod) and typed in the command to select one leg, then typed the command to raise it until it looked right. Having a 32-bit processor like the 68332 as the master CPU was a joy. Not only is the 68332 a useful processor for controlling devices (because of the large number and flexibility of its I/O lines), but it has the computational power to do sophisticated processing and still satisfy the real-time needs of controlling a robot.

## The Motorola 68332

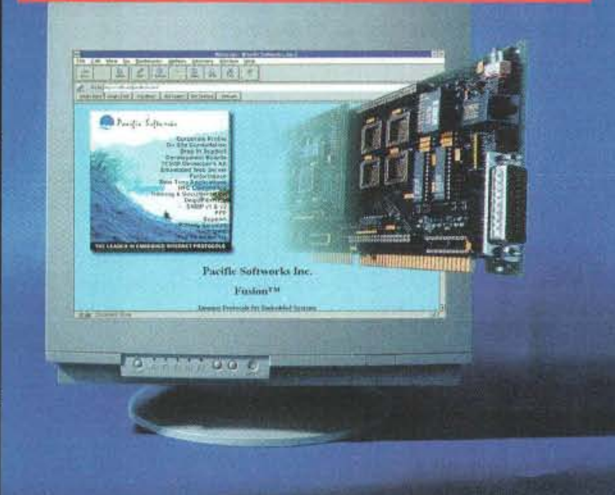
The short description of the Motorola 32-bit 68332 microcontroller is that it is an updated 68000 processor with several integrated coprocessors for memory management and I/O support. The two most important coprocessors are the Queued Serial Module (the QSM) and the Time Processor Unit (the TPU).

The QSM provides a pair of asynchronous serial I/O lines and several SPI synchronous I/O lines. Using the asynchronous I/O is much like having a UART on chip: You set a few configuration registers, which control such things as the baud rate, parity, and so on, and then serial I/O can be run in the background with

- FULL HTML SUPPORT
FUSION Web Server supports all popular browsers and HTML levels 2.0, 3.0 and 3.2 including Netscape extensions so that your pages can present information effectively.
- SNMP-Style Interface
An SNMP-style interface provides powerful access to set and get routines.
- SESSION PROTECTION REALMS
Allow sensitive pages to be protected in groups rather than requiring a password per page.
- CONVERSION PROGRAM
The Included Conversion program preprocesses and compresses your Web pages into compilable code, so that the pages may be developed using standard HTML tools.
- MIME SUPPORT
MIME types supported include HTML pages, standard pages and Java applets.
- DYNAMIC HTML SUPPORT
Dynamic HTML creation includes support for index-based variables and Netscape-style dynamic page support automatically updates pages.
- COMPILATION OPTIONS
Compilation options include buffer sizing, image map support, MIME type checking, clock support, session timeouts, index depth, and password groups.

THE LEADER IN EMBEDDED INTERNET PROTOCOLS

FUSION WEB SERVER



WHEN TIME TO MARKET IS KEY:
THE FUSION EMBEDDED WEB SERVER ALLOWS YOUR EMBEDDED SYSTEM TO COMMUNICATE WITH ANY WEB BROWSER SUCH AS NETSCAPE, EXPLORER OR MOSAIC. YOU CAN TURN YOUR EMBEDDED SYSTEM INTO A WEB SERVER, ELIMINATING THE EFFORT OF WRITING A CUSTOMIZED APPLICATION FOR CONTROL AND MANAGEMENT.

PACIFIC SOFTWORKS PROVIDES ONGOING SUPPORT OF THE EVOLVING HTTP/HTML SPECIFICATIONS ENABLING OUR CUSTOMERS TO ALLOCATE ENGINEERING RESOURCES TO OTHER CRITICAL TIME TO MARKET TASKS.

OUR "TIME TO MARKET PHILOSOPHY" HAS MEANT WE'VE DEVELOPED PRODUCT WITH "DROP-IN SUPPORT" FOR WEB SERVER APPLICATIONS, ETHERNET AND PROCESSOR CHIPS AND REALTIME KERNELS. HIGH QUALITY, HIGH PERFORMANCE INTERNET APPLICATIONS DEVELOPED WITH OVER 14 YEARS OF INTERNET PROTOCOL EXPERIENCE FOR YOUR SYSTEM DESIGNS.

New Version!

Pacific Softworks

*FOR MORE INFORMATION CALL:*
*(800) 541-9508 within USA*
*(805) 484-2128 outside USA*
*(805) 484-3929 FAX*
*Internet: sales@pacificsw.com*
*Europe, Middle East and Africa*
*Ph: (44) 1494-432-735*
*Email: sewellm@pacificsw.com*
*FAX: (44) 1494-432-728*
*OR VISIT OUR WEBSITE:*
*World Wide Web: http://www.pacificsw.com*

CIRCLE NO. 162 ON READER SERVICE CARD

no need for direct management from the CPU. The SPI section of the QSM implements Motorola's bit-synchronous I/O system. This is a three-wire (Master-In, Master-Out, and Clock) system that is supported by many devices such as A/D chips. There are significantly more registers to set in order to use SPI, because of the large variety of possible configurations. The data-transfer protocol is different, but conceptually, the configuration mechanism is same as the asynchronous I/O: You configure the subsystem by setting the registers and then the I/O occurs in the background. Without adding external circuitry, up to four SPI devices can be used; a little bit of select decoding logic allows 16 devices to be used. For more details on the QSM, see "Time for the 68332," by Eric McRae (*DDJ*, January 1995).

The TPU is an extremely powerful I/O subsystem. It provides up to 16 I/O lines running up to 10 different I/O functions. Many of these functions involve timing in one form or another. The TPU is a bit intimidating to learn to use. It takes writing half-a-dozen bytes just to set up to do a simple toggle of an I/O bit. But you can also write half-a-dozen bytes to set up period measurement with additional transition detection, or a stepper-motor controller with programmable acceleration and deceleration. Using the interpreter and incremental compilation features of Forth on the New Micros single-board 68332 system makes learning to use the TPU much easier. The TPU is described in more detail in "Inside Motorola's TPU," by Richard Soja (*DDJ*, December 1996).

## Finite-State Machines

Since the control software for a robot operates on many different levels, it needs to be well factored, or it quickly becomes unmanageable. For small systems, a finite-state machine is a natural way to partition the problem. This is a conceptualization of the system as always being in one of several states. The system makes well-defined moves from one state to another in response to inputs or events. For example, consider the state machine in Figure 1 that describes how the robot will respond to bumping into something. In this machine the states are:

- BUMP_SCAN when there is no contact.
- LEFT_BUMP when the left whisker makes contact (by design, the robot treats the situation where both whiskers make contact as a left contact).
- RIGHT_BUMP when the right whisker makes contact.

There is an additional state, END_BUMP, which the robot enters when a left or right contact ends. The END_BUMP state exits back to BUMP_SCAN after a timeout period has expired. This gives the robot the time to turn away from the obstruction. In the strictest computer-science

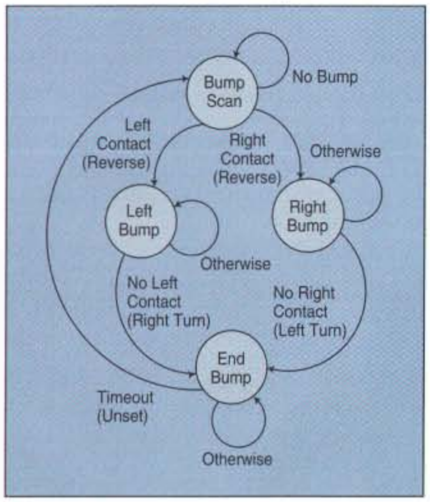


***Figure 1:** An augmented finite-state machine to manage collisions between the robot and an object. The action that the robot takes upon exiting a given state (in the circles) is in the parentheses for each of the transition paths.*

NEW EMBEDDED WEB SERVER
Use FUSION Web Server as a powerful graphical application designed for control and management applications using standard HTML forms. Works with all Web browsers.

NEW DEVELOPMENT BOARDS
Use FUSION Development Boards to test drive FUSION TCP/IP and application software, or develop your own application on these powerful platforms.

NEW ISDN PC DIAL-UP BOARDS
ISDN ISA and PCMCIA adapters used for high speed dial-up connection to your service provider.

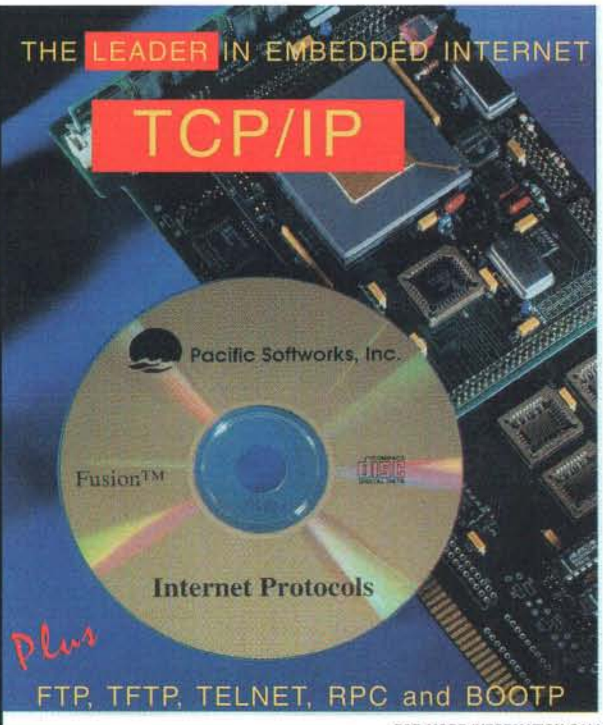
NEW HIGH SPEED ROUTING
RIP & OSPF protocols for high speed embedded system routing.

SNMP V1, V2
SNMP MIB COMPILER
High Performance SNMP for embedded systems. Supports v1 & v2 parameters with a powerful MIB compiler.

PPP, SLIP, CSLIP
PPP is used for most serial links. Allows for transfer of data over a serial link with authentication.

THE LEADER IN EMBEDDED INTERNET PROTOCOLS

TCP/IP



Plus
FTP, TFTP, TELNET, RPC and BOOTP

WHEN TIME TO MARKET IS KEY: USE FUSION: TCP/IP, SNMP, PPP, DEVELOPER KITS ALONG WITH OUR EMBEDDED WEB SERVER, DEVELOPER "SURFBOARDS" AND ISDN SOLUTIONS.

FUSION DEVELOPER KITS ARE THE MOST POWERFUL INTERNET PROTOCOLS DEVELOPED FOR EMBEDDED APPLICATIONS. OUR "TIME TO MARKET PHILOSOPHY" HAS MEANT WE'VE DEVELOPED PRODUCT WITH "DROP-IN SUPPORT" FOR WEB SERVER AND ISDN APPLICATIONS, ETHERNET AND PROCESSOR CHIPS AND REALTIME KERNELS. HIGH QUALITY, HIGH PERFORMANCE INTERNET APPLICATIONS DEVELOPED WITH OVER 14 YEARS OF INTERNET PROTOCOL EXPERIENCE FOR YOUR SYSTEM DESIGNS.

**Introducing the Pacific Softworks**
SURF SHOP
Product line of development, evaluation and application boards.

Pacific Softworks

*FOR MORE INFORMATION CALL:*
*(800) 541-9508 within USA*
*(805) 484-2128 outside USA*
*(805) 484-3929 FAX*
*Internet: sales@pacificsw.com*
*Europe, Middle East and Africa*
*Ph: (44) 1494-432-735*
*Email: sewellm@pacificsw.com*
*FAX: (44) 1494-432-728*
*World Wide Web: http://www.pacificsw.com*

CIRCLE NO. 340 ON READER SERVICE CARD

definition of a finite-state machine, timer events (like timeouts) are not allowed; augmented finite-state machines that do recognize timer events, however, are common in process-control systems.

### The IsoMax system

The New Micros IsoMax system is a finite-state-machine engine that sits on top of a Forth compiler. Typically you write the low-level control code in either assembler or Forth, and the high-level system as a set of interacting state machines. IsoMax internally provides the glue to drive the system through the states and to run all the state machines. You can then concentrate on describing the application specifics and not worry about building the state-machine engine itself.

Using IsoMax is straightforward. You basically write a specification of all the state machines. There are four fundamental sections of an IsoMax state-machine specification:

- The list of states that apply to a given state machine. Example 1(a) defines the machine PROXIMITY to consist of five different states.
- For each state, give the conditions that cause a transition to another state. Example 1(b) sets up the response of the PROXIMITY machine when it is in the state IR_SCAN. If the word BOTH_IR? returns a True condition, then the sequence of Forth words between CAUSES and THEN-STATE run (which causes the robot to back up). After this, the state switches to BOTH_IR. The machine can have as many of these clauses as necessary. Only the conditions that must be acted upon need to be listed in the IN-STATE...TO-HAPPEN clauses; all other conditions leave the machine in the current state.
- Next, all the defined state machines are linked into a single list of machines, which make up a composite-state machine that is to be run (Example 1(c)). In IsoMax, no priority is implied by the order of the machines in this list. Priorities or subsumption overrides must be built into the supporting words. In my robot, the state transitions cause a word that I wrote, ARBITRATE, to be called either directly or indirectly. This simple word chooses one of several possible actions (set by each state machine) on a priority basis.
- Finally, before actually running the machine chain, the initial state of each of the state machines must be defined; see Example 1(d).

```
(a)  STATE-MACHINE PROXIMITY
        APPEND-STATE IR_SCAN
        APPEND-STATE LEFT_IR
        APPEND-STATE RIGHT_IR
        APPEND-STATE BOTH_IR
        APPEND-STATE END_IR

(b)  IN-STATE IR_SCAN
        CONDITION BOTH_IR?
        CAUSES PROXIMITY_VEC
          SET_REVERSE ARBITRATE
        THEN-STATE BOTH_IR
     TO-HAPPEN

(c)  MACHINE-CHAIN ROBOT
         SENSORS
         WALKER
         CONTACT
         PROXIMITY
         EXPLORE
     END-MACHINE-CHAIN

(d)    WALK_FORWARD SET-STATE
       IR_SCAN      SET-STATE
       BUMP_SCAN    SET-STATE
       IDLE         SET-STATE
       IDLE_MOTION  SET-STATE
```

***Example 1:*** *(a) The PROXIMITY machine; (b) setting up the response of the PROXIMITY machine; (c) linking state machines into a single list of machines, which makes up a composite-state machine; (d) defining the initial state of each state machine.*

**IPX Routing**
RIP/SAP
NLSP
IPXWAN
SPX

**IP Routing**
TCP
OSPF
SNMP
DHCP
BOOTP
DNS
TELNET
TFTP

**Bridging**
SR
STP
NetBIOS G/W

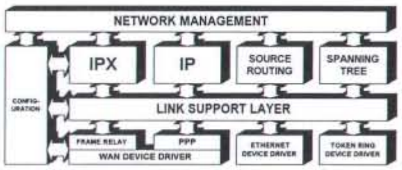
**WAN**
PPP
Multilink
Compression
Frame Relay
DLSw

**& More...**

**Coming Soon**
RSVP
Multicast IP
PPTP

## Portable Protocol Technology

If you are manufacturing an internetworking product and are in need of protocol software, consider using **ROUTER*Ware*** as your source of technology. Our family of royalty free ANSI C based products have been used by more than 70 vendors world wide to meet ever diminishing time to market and budget requirements.



Contact us for further information at:

**ROUTER*Ware***

3961 MacArthur Blvd., Suite 212
Newport Beach, California 92660 USA

**(714) 442-0770 Fax: (714) 442-0779**

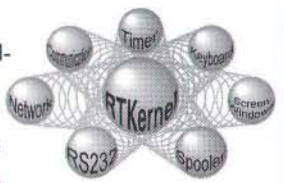
Send inquiries to **info@routerware.com** or visit us at **http://www.routerware.com**

CIRCLE NO. 331 ON READER SERVICE CARD

## Real-Time Multitasking Tools for Embedded Systems and DOS

**RTKernel**

Professional, high-performance real-time multitasking system for DOS and 16-bit Embedded Systems.
For Borland C/C++, Microsoft C/C++, and Borland Pascal.
**Libraries: $550 Source Code: add $500**



**RTTarget-32**

Cross Development System for 32-bit Embedded Systems.
Supports Intel 386 and higher, as little as 16k RAM/ROM.
For Borland C/C++, Microsoft C/C++, and Watcom C/C++.
**Libraries: $1700 Source Code: add $1000**

**RTKernel-32**

Professional, high-performance real-time multitasking system for 32-bit Embedded Systems.
Supports Intel 386 and higher.
For Borland C/C++, Microsoft C/C++, and Watcom C/C++.
**Libraries: $1950 Source Code: add $1650**

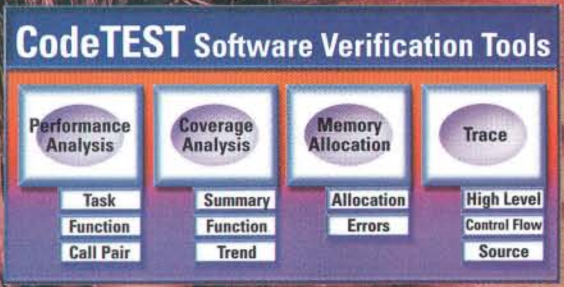
**In North America, please contact:**
On Time
88 Christian Avenue, Setauket, NY 11733, USA
Phone (516) 689-6654
Telefax (516) 689-1172
email info@on-time.com

**Other Countries:**
On Time Informatik GmbH
Hofweg 49, 22085 Hamburg, GERMANY
Phone +49-40-2279405
Fax +49-40-2279263
email 102212.3101@compuserve.com

**NO ROYALTIES! FREE DEMO DISK!**
**http://www.on-time.com**

On Time
REAL-TIME AND SYSTEM SOFTWARE

# GET A GRIP.



Two weeks to release date and you're still finding bugs. Your boss is breathing down your neck. Can you do it? No problem. Now you can perform unit, integration and system testing, quickly and reliably. New CodeTEST™ embedded software verification tools from Applied Microsystems. Get answers fast. And get a grip.

Call Now 1-800-895-0831
or visit our home page at http://www.amc.com
for free literature and informative screen shots.

Applied Microsystems Corporation

CIRCLE NO. 338 ON READER SERVICE CARD

*(continued from page 54)*

## Subsumption

Implementing a complete robot-control system as a single finite-state machine rapidly gets unwieldy as new systems are added. This is especially true for research robots that may have radically different sets of sensors and mechanical systems each time they are powered up. A more manageable approach is to create multiple finite-state machines. You implement each subsystem as a finite-state machine, then set up an arrangement that clearly defines how these separate state machines will interact.

One approach to defining how the state machines should interact is the subsumption architecture. The subsumption architecture was proposed for robotic control applications by R.A. Brooks in "A Robust Layered Control System for a Mobile Robot" (*IEEE Journal of Robotics and Automation, Vol. RA-2, No. 2,* 1986). It has become a popular scheme for robots because it is flexible, easy to manage, scales well for complicated robots, and degrades gracefully in the case of sensor failures. It can produce surprisingly complicated behaviors from small programs, particularly when the robot is trying to simultaneously achieve multiple goals.

The idea is that each state machine uses the relevant inputs (from sensors and internal states) and continuously generates an output of control messages. Then, downstream of the control messages and before the actual hardware, is an arbitration mechanism. The arbitrator can be designed in several ways, but it can be thought of as a series of enable and inhibit nodes (similar to the way that actual neurons are interconnected; also like real neurons, it is the inhibit connections that play the major role). These nodes are interconnected in a way that ultimately results in only one of the multiple incoming control messages actually getting out to the hardware layer.

Figure 2 illustrates the state machines for this robot:

**SENSORS** runs all the robot sensors and sets all relevant global variables. This machine also initializes all the subsystems when the robot's "go" switch is turned on, and stops the robot when the switch is turned off. Unlike all the other state machines, SENSORS is not really part of the subsumption scheme. Instead, this machine provides a central background task that assures all the sensors are scanned. The IR and contact sensors are essentially continuously scanned. Once every six seconds, the sonar is used to look around for obstacles.

**WALKER** sequences through all six robot legs in the currently defined sequence. The current sequence is determined by six execution vectors, each one defining which leg to move and the type of cycle to execute for it. The default sequence is to move straight forward, but it can be overridden. This machine also has an "idle" state that is entered at the end of the sixth-leg sequence, before going back to the first leg. Usually the machine will just transit from the sixth sequence to idle to the first sequence. But a general robot-shutdown message will cause this machine to hold in the idle state until a "go" message arrives, at which point it will

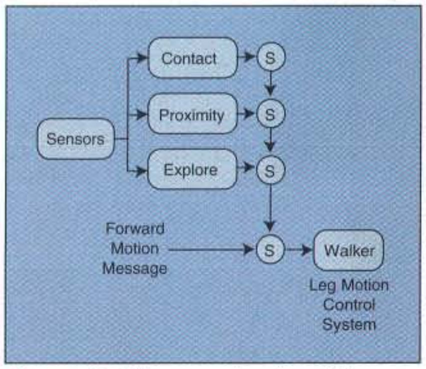


***Figure 2:*** *The organization of the multiple state machines (in the boxes) for the robot in a subsumption scheme. Motion-control messages coming into the "S" nodes vertically will override a message coming into the node from the horizontal direction. Note that in the absence of any overriding messages, the robot will walk straight forward.*

**More Bang For Your Bug**

*Everything you need for embedded 80x86 development*

**Fast and easy-to-use debugging** — essential when developing your application. CSi's Soft-Scope is a source-level debugger that blows away pesky bugs while fully testing and integrating your system.

**Versatility is key** — Compatible in real or protected mode with Intel, AMD, National Semiconductor, TI and NEC 80x86 processors. Integrated with Microsoft, Borland, Watcom and MetaWare compilers.

**Now with kernel awareness, linker and GOFAST® floating point** — Uses the industry-standard debug interface supported by SingleStep™ and Soft-Scope.

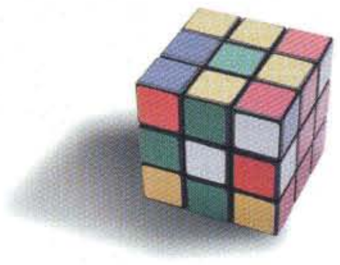
**Work in Win95, WinNT, or Win3.1** — Now you can utilize your preferred application development platform.

Soft-Scope comes with the same reliable performance and support you've come to expect from all of CSi's debugging tools. Give us a call at **800-897-3001** and find out how we can help make your life bug-free.

P.O. Box 9666, Moscow, ID 83843 USA
(USA) 800-897-3001 • 208-882-0445
Fax: 208-882-9774 • Email: info@consci.com
**http://www.debugger.com**

**Concurrent Sciences, Inc.**

CIRCLE NO. 79 ON READER SERVICE CARD



**Par•a•digm:** your source for the most high-powered, comprehensive set of time-saving software and hardware development tools for embedded application development.

1: **Paradigm LOCATE** the most popular tool for creating embedded C/C++ applications with Borland and Microsoft compilers; **2: Paradigm DEBUG** the only x86 debugger with C++, RTOS, scripting language, and full in-circuit emulator support; **3: Paradigm SUPPORT** the best technical support in the industry supplied to our customers for free.

Developing real-time embedded applications doesn't have to be time consuming or difficult—you just need to have the right tools. **Paradigm** alone has the high performance development tools you need to streamline the embedded system software development process so your Intel and AMD x86 applications are ready in record time. Paradigm's complete suite of tools work with industry standard C/C++ compilers from Borland and Microsoft, as well as hardware development tools from Applied Microsystems, Beacon Development Tools and other popular in-circuit emulator vendors.

Call us at **800-537-5043** today and let us take care of all your development tool needs, so you can keep your focus where you need it—on your application.

Paradigm

**Paradigm Systems**
3301 Country Club Road
Suite 2214
Endwell, NY 13760

**1-800-537-5043**

Phone **607-748-5966**
Fax **607-748-5968**
**info@devtools.com**
**http://www.devtools.com**

86

Desperately seeking DOS for fully compatible LTR. Flexibility and good memory desired. Need not be extravagant. Must be good embedded. Free demo 1-800-221-6630

SWM, enjoys computer games

**ROM-DOS™**

Find your fully compatible DOS and save up to 80% compared to MS-DOS®. It even runs Windows™.

That number again is
**1-800-221-6630**

**Datalight®**

*System Software for Bright Ideas*

**18810 59th Ave. NE • Arlington, WA 98223**
**(360) 435-8086 • FAX: (360) 435-0253**
**e-mail: sales@datalight.com**
**www: http://www.datalight.com**

ROM-DOS is a trademark of Datalight.
*MS-DOS and Windows are registered trademarks of Microsoft Corp.

**CIRCLE NO. 151 ON READER SERVICE CARD**

```
: ARBITRATE
    ( this sequence gives a subsumption behaviour to the system )
    CONTACT_VEC   @ DUP IF EXECUTE EXIT ELSE DROP THEN
    PROXIMITY_VEC @ DUP IF EXECUTE EXIT ELSE DROP THEN
    EXPLORE_VEC   @ DUP IF EXECUTE EXIT ELSE DROP THEN
    WALK_VEC      @ DUP IF EXECUTE EXIT ELSE DROP THEN
    ;
```

***Example 2:*** *The arbitrator.*

*(continued from page 56)*
cycle through all the legs again (with a walking robot, it is a good idea to have a well-defined way to stop walking, or it might fall over).

**EXPLORE** makes the robot wander around in open space until the sonar determines that it is getting close to an obstruction. It then looks for the largest possible open region to move into. The sonar is swept across the field of view of the head in order to find out what is the direction of the largest range. The direction that EXPLORE decides to move can determine what six-leg sequence that WALKER will use.

**PROXIMITY** decides upon a course of action based on the IR system's detection of an obstacle in the vicinity. If an object is detected nearby, the robot turns to walk away from it. This machine can override the sequence that WALKER would get from EXPLORE.

**CONTACT** decides upon a course of action based on the contact whiskers' detection of an object in contact. The behavior here is similar to PROXIMITY, except that since the robot is in contact with the object, it backs up several steps before turning away from the object. This machine can override both EXPLORE and PROXIMITY.

During development and testing, you cannot just release a half-dozen asynchronous, interacting state machines and expect to see a working robot. When designing a system that contains multiple state machines, it is important that the individual machines interact with each other in a restricted and well-defined manner. This makes it easier to develop and test each subsystem, since you can work with a single state machine at a time. It also keeps the coding simpler, since the complications of semaphores or other access synchronization schemes are avoided. In this robot, for example, only the state machine WALKER actually moves the legs. The other machines communicate with WALKER by providing the desired sequences of leg movements. This communication is handled in the arbitration code. Once the individual state machines are properly running, the overall control system can be tested by adding one state machine at a time.

For all its importance, the arbitrator for managing the state machines is a small routine. It chooses one of several possible execution tokens (function pointers) and runs it. Example 2 is the entire arbitrator. This Forth word looks in turn at the execution vectors that may be set by the machines CONTACT, PROXIMITY, EXPLORE, and WALKER. The order in which the vectors are tested sets the priority order. The first non-null vector it encounters is executed, then the arbitrator exits. When run, each of these routines set the six execution vectors for the gait sequence used by WALKER, causing the robot to start walking with the leg sequence determined by the state machine with the highest priority. Under ordinary circumstances, WALK_VEC contains a pointer to the word that defines the gait to forward. This means that the default behavior of the robot is to walk forward, unless it is overridden by a high-priority state machine.

## The Future

One generalization about robots that holds true is that they are always work in progress. For quite some time I was preoccupied with just getting this one to walk at all. Now I am focusing on the real-time process-control-level code that defines the robot's basic behavior (this is what I have described here).

In the future, there are higher-level problems to solve that fall into the domain of artificial intelligence. For example: Given a map of a room, how does a robot placed in an arbitrary location in that room determine where it is? Even more challenging is the question of how it locates itself without a map. Robots certainly are not boring to work with: There is always a new challenge.

**For More Information**

New Micros Inc.
1601 Chalk Hill Road
Dallas, TX 75212
214-339-2204
http://www.newmicros.com/

**DDJ**

# We'll Catch You On The Web.

**Visit us at http://smallest.pharlap.com to see how you can develop embedded applications that take full advantage of the web.**

Phar Lap's TNT Embedded ToolSuite® Realtime Edition, complete with Realtime ETS™ Kernel, now comes with our robust networking protocol—ETS TCP/IP! This cutting edge feature allows your customers' mainframes, workstations, or PCs to communicate with products on the factory floor, in the lab, or at remote sites—all using Web technology!

As a result, electronic OEMs can use this technology to create intelligent machines and instruments for an unlimited number of applications such as medical instruments, robotics, avionics equipment etc.

In addition, we support a variety of network protocols as well as industry standard tools such as Visual C++, Borland C++, CodeView, and Turbo Debugger.

And because Phar Lap's TNT Embedded ToolSuite comes with development tools, a Realtime ETS Kernel, and now ETS TCP/IP, it is a one-stop shopping product with enormous power!

To find out more, catch us on the "World's Smallest Web Server" today at URL http://smallest.pharlap.com or call a Phar Lap sales representative! You'll be surprised how powerful the Web can be.

The 32-Bit x86 Experts

PHAR LAP

Embedded Development — Simply on Target™

Phar Lap Software, Inc. 60 Aberdeen Avenue, Cambridge, MA 02138 • Tel: (617) 661-1510 • Fax: (617) 876-2972 • http://www.pharlap.com

CIRCLE NO. 99 ON READER SERVICE CARD

# UNIX Filesystems without I-Nodes

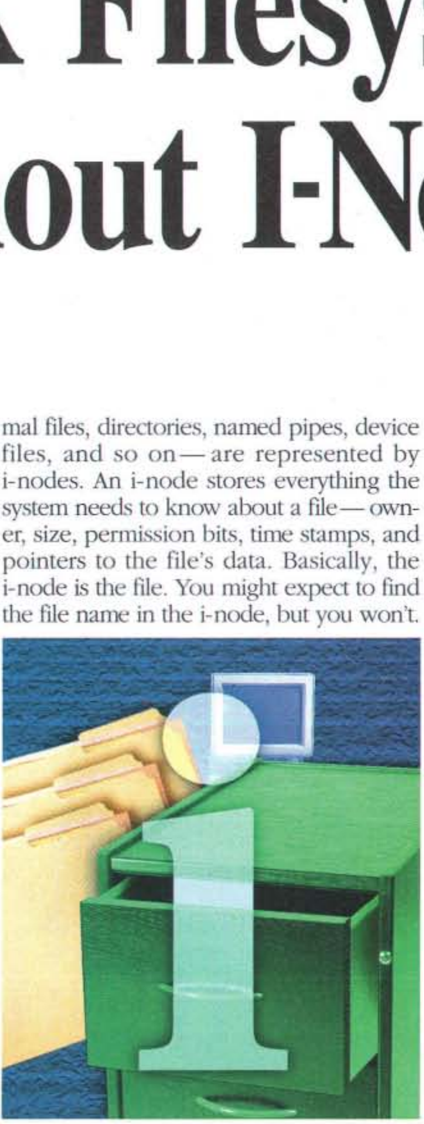
*NCP and SMP support in Linux*

Volker Lendecke

Along with nfs, the Linux kernel smbfs and ncpfs filesystems make it possible to link Linux machines to virtually any file server—from Pathworks to Windows NT 4.0, from NetWare to any NFS server—across a LAN. When I was implementing smbfs and ncpfs, however, it became clear that Microsoft's Server Message Block (SMB) protocol is not designed to handle UNIX clients like Linux. SMB, the protocol that implements file services, is designed for DOS. Consequently, SMB has no notion of an i-node, the central structure in every UNIX filesystem implementation. On the surface, this would appear to limit Linux's usefulness on heterogeneous networks. However, in this article, I present techniques I developed to work around this limitation.

### UNIX Filesystems

The user's view of a UNIX filesystem is straightforward: Everything is arranged in a hierarchy of directories. But this hierarchical order does not reflect the filesystem's layout on disk. In reality, all things that can be stored in a filesystem—normal files, directories, named pipes, device files, and so on—are represented by i-nodes. An i-node stores everything the system needs to know about a file—owner, size, permission bits, time stamps, and pointers to the file's data. Basically, the i-node is the file. You might expect to find the file name in the i-node, but you won't.



The name is stored elsewhere. I-nodes are arranged in a large array stored at the beginning of a disk partition, or scattered around within the partition. The filesystem can only reference i-nodes via their numbers, the indexes into the i-node table.

The illusion of a hierarchical filesystem is created by the directory, a special type of file. On disk, directories look much like normal files; they have an i-node and data bytes. Their data simply consists of a list of filenames paired with i-node numbers. These pairs of names and i-nodes are what users see as files. The difference between normal files and directories is that users are not allowed to access the directories. If a program manipulated directories, the UNIX system could not maintain its hierarchical structure. This loose coupling between a file's name and the file itself is the reason you can have two different names for a single file. You just create two directory entries pointing to the same i-node; the names can even be in different directories. Moving files in the filesystem is also simple: Create a filename with the file's i-node number in the target directory and delete the file's entry in the source directory. In short, a file gets an i-node number when it is created. This i-node number remains constant during the lifetime of the file, no matter where you move it in the directory tree.

This central role of i-nodes can also be seen in the implementation of the Linux virtual filesystem (vfs). In fact, the i-node structure (see Figure 1) is the most important object in the Linux vfs. An i-node is read from a table on disk when the corresponding file is opened the first time, or when the user wants information about the file. For example, the command *ls – l* reads every file's i-node and shows the metadata of each file it lists. The i-node is written back when the last process that accessed the file closes it. All other operations that can be performed on a file, including read/writes, refer to the i-node.

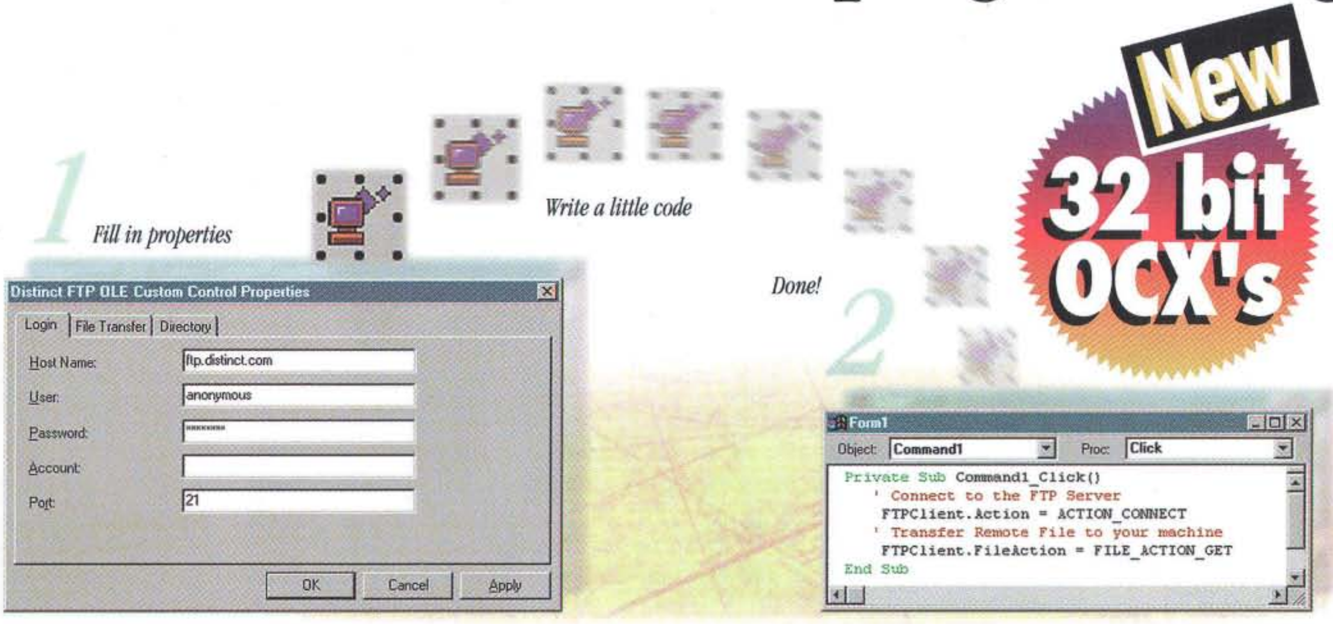
### Drawbacks for Linux

Both Microsoft's SMB protocol and early versions of Novell's NetWare Core Protocol (NCP) were designed to redirect all file-access functions of the DOS INT 0x21 to the server. The native DOS filesystem (that is, the FAT system) has no notion of i-nodes. FAT does not separate filenames from the files themselves. The meta data of the files is stored in the directory the file resides in together with its name. Because no i-nodes or i-node numbers are

*Volker, who implemented the ncpfs and smbfs filesystems for the Linux kernel, studies mathematics and computer science at the University of Göttingen, Germany. He can be contacted at lendecke@math .unigoettingen.de.*

# Visual Internet™ Toolkit

## Save months of TCP/IP programming!



| | |
|---|---|
| Host Name: | ftp.distinct.com |
| User: | anonymous |
| Password: | ******** |
| Account: | |
| Port: | 21 |

Object: Command1 Proc: Click

```
Private Sub Command1_Click()
    ' Connect to the FTP Server
    FTPClient.Action = ACTION_CONNECT
    ' Transfer Remote File to your machine
    FTPClient.FileAction = FILE_ACTION_GET
End Sub
```

## OLE Custom Controls

Just ask any OCX jockey. With Distinct's Visual Internet Toolkit, adding TCP/IP connectivity to your application is not much farther than a drag-and-drop away. Whether you need a customized FTP client or most any other Internet application, you can simply embed an OCX into your program and ***Visual Internet will do the rest.*** It's that easy. And you'll have great looking, powerful applications.

## 32 Bit Performance

The power of our new 32 bit Visual Internet Toolkit is simply unsurpassed. More custom controls. More protocols. More sample code. More Documentation. Which makes your job easier and leaves the competition in the dust.

### Protocols

- Windows Sockets
- TCP/UDP/ICMP
- PPP/SLIP/CSLIP
- E-mail/SMTP
- POP 2/POP 3
- News/NNTP
- FTP
- TFTP
- TCP Server
- Telnet
- VT 220
- WinSNMP
- ONC RPC/XDR
- rcp
- rexec
- rlogin
- rsh
- And many more

| Interfaces* | Environments |
|---|---|
| • 32 bit (95 and NT) | • Visual Basic |
| • 16 bit (Windows 3.x) | • Visual C/C++ |
| • C++ Class libraries | • Delphi |
| • DLL's | • C/C++ |
| • OCX's | • Access |
| • VBX's | • FoxPro |

***Call now for 30 minute Internet Delivery!***

distinct®

*The world leader in Internet development tools.*

**408.366.8933**

| | |
|---|---|
| **World Wide Web:** | **http://www.distinct.com** |
| **Fax:** | **408.366.0153** |
| **E-mail:** | **drdobbs@distinct.com** |
| **Fastfacts:** | **408.366.2101** |

*Not all interfaces may be available for all protocols. Licensing fees required for redistribution. Distinct is a registered trademark and 30 minute Internet Delivery! and Visual Internet is a trademark of the Distinct Corporation. Copyright 1995 Distinct Corporation, 12900 Saratoga Avenue, Saratoga, CA 95070. All rights reserved. Specifications and delivery terms are subject to change without notice.

**CIRCLE NO. 162 ON READER SERVICE CARD**

(continued from page 60)
needed to implement the INT 0x21 interface, neither SMB nor NCP provide i-node numbers. All that is transferred over the network is the pathname of a file. In NCP, you can allocate 1-byte handles for parts of the path. But these handles cannot be used as i-node numbers, as they are too short and do not uniquely identify a file.

To implement a filesystem for Linux, you have to implement *read_inode()*. The upper layers of the Linux filesystem (implemented in the linux/fs/*.c source files) hand an i-node number to this function and expect the actual file-system implementation to fill a prepared i-node structure with the values the i-node has on disk. But what do you do when you can't ask the file server to give you "the size of the file with i-node 1234" because the server doesn't know anything about i-node 1234? That's the problem I faced when implementing a filesystem for Linux. Clearly, I had to find a way to fool the Linux kernel.

When writing smbfs and ncpfs, I had to identify all points in the Linux vfs layer that rely on the fact that a file is essentially an i-node with a fixed i-node number. I did not expect to find only two vfs routines to deal with i-node numbers: *lookup()* and *read_inode()*. (*create()* is a third, but it works exactly as *lookup()* does.) The kernel asks *lookup()* for the i-node of a file, giving it the filename and the directory to search. *lookup()* finds the i-node number of the specified file, then calls *iget()* to read the i-node. *iget()* is a higher-level function that maintains a cache of i-nodes. When it does not find the requested i-node in its cache, it calls *read_inode()* with the i-node number it just got from *lookup()*.

So smbfs and ncpfs have to make sure that *lookup()* generates unique i-node numbers for all the i-nodes *iget()* holds in memory. And *lookup()* needs a way to tell *read_inode()* about the i-node numbers it has generated artificially. The Linux kernel doesn't care about the i-node number stored on disk or anywhere else. The i-nodes in memory are those that are referenced as open files from one or more processes and are the current working directory of a process. The numbers of all other i-nodes can be randomly chosen. The only problem that remains is determining how the filesystem should choose the i-node numbers.

One solution is to generate them randomly. However, this does not provide uniqueness. Another idea is to increment a counter whenever an i-node number is requested. But as Linux systems are expected to run for a long time and must therefore cope with wraparound, a list of numbers in use must be maintained. This is inefficient. The fastest solution turns out to be the simplest: The Linux kernel already has an extremely reliable number generator perfectly suited for this purpose—*kmalloc()*.

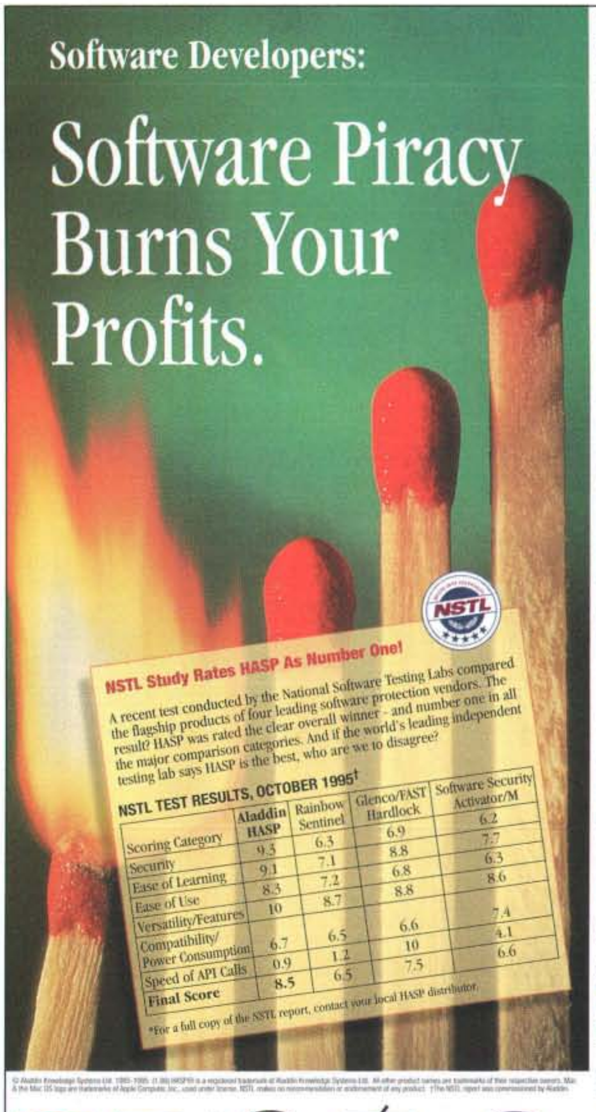
## The *kmalloc()* Approach

Normally, when a file-system type is added to Linux, the global i-node structure is augmented by the fields that are needed by the new type. This can be found in the file linux/include/fs.h. When I was developing smbfs on the stabilized kernel 1.2.13, this was not an option, because I wanted to run these filesystems as modules without kernel modifications. Additionally, the structures needed by smbfs and ncpfs are quite large, and I did not want to penalize all other filesystems with such a large structure. So I chose to store the smbfs/ncpfs-specific i-node data in a special *kmalloc()* structure. Now it's very fast to find unique i-node numbers: I simply use the address of the allocated structure. We know this number is unique across all filesystems and it references the corresponding structure without overhead.

I-node numbers are generated in the vfs function *lookup()*. This function allocates the special file-system-specific structure and gives its address as the i-node number to the kernel routine *iget()*. *iget()* finds that the i-node is not in memory yet and calls *read_inode* with the wanted number. *read_inode* is now able to find the file-system-specific data by a type cast. The routine *put_inode* can reliably *kfree*

**Software Developers:**

**Software Piracy Burns Your Profits.**



**NSTL Study Rates HASP As Number One!**

A recent test conducted by the National Software Testing Labs compared the flagship products of four leading software protection vendors. The result? HASP was rated the clear overall winner - and number one in all the major comparison categories. And if the world's leading independent testing lab says HASP is the best, who are we to disagree?

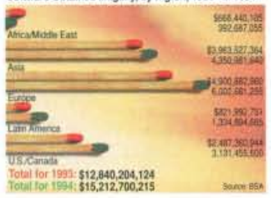
**NSTL TEST RESULTS, OCTOBER 1995**†

| Scoring Category | Aladdin HASP | Rainbow Sentinel | Glenco/FAST Hardlock | Software Security Activator/M |
|---|---|---|---|---|
| Security | 9.3 | 6.3 | 6.9 | 6.2 |
| Ease of Learning | 9.1 | 7.1 | 8.8 | 7.7 |
| Ease of Use | 8.3 | 7.2 | 6.8 | 6.3 |
| Versatility/Features | 10 | 8.7 | 8.8 | 8.6 |
| Compatibility/ Power Consumption | 6.7 | 6.5 | 6.6 | 7.4 |
| Speed of API Calls | 0.9 | 1.2 | 10 | 4.1 |
| **Final Score** | **8.5** | 6.5 | 7.5 | 6.6 |

*For a full copy of the NSTL report, contact your local HASP distributor.

Each year, the illegal use of software consumes nearly 50% of your potential revenues. With the flames of piracy eating away at your profits, can you afford not to protect your software? HASP® is widely acclaimed as the

Software Obtained Illegally, by region, 1993 vs. 1994



world's most advanced software protection solution. Since 1984, thousands of leading developers have used nearly two million HASP keys to protect billions of dollars worth of software.

Why? Because HASP's security, reliability, and ease-of-use led them to a simple conclusion: HASP is the most effective software protection system available.

Today, more developers are choosing HASP than any other software protection method. To learn why, and to see how easily you can increase your revenues, call now to order your HASP Developer's Kit.

**1-800-223-4277**

http://www.aks.com

1985-1995 10 YEARS

ALADDIN

*The Professional's Choice*

| | |
|---|---|
| North America | **Aladdin Software Security Inc.** Tel: (800) 223-4277, 212-564-5678 Fax: 212-564-3377 E-mail: sales@hasp.com |
| Intl Office | **Aladdin Knowledge Systems Ltd.** Tel: 972-3-537 5795, Fax: 972-3-537 5796 E-mail: sales@aks.com |
| United Kingdom | **Aladdin Knowledge Systems UK Ltd.** Tel: 01753-622266, Fax: 01753-622262 E-mail: sales@aldn.co.uk |

Please call for details of your local distributors

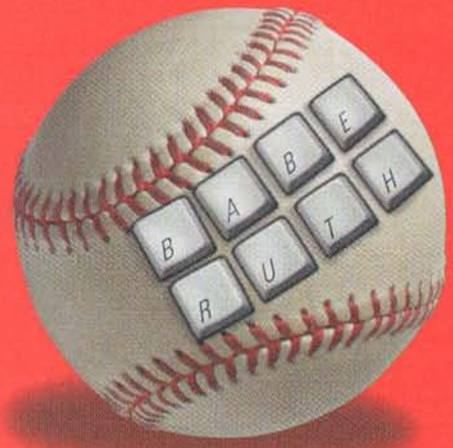
yes It runs with NetWare Developer tested only

Available and Ready for OS/2

MICROSOFT® WINDOWS™ COMPATIBLE

Mac™ OS

CIRCLE NO. 167 ON READER SERVICE CARD

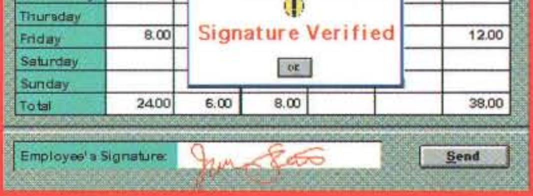


# Some things aren't worth much without a signature.

You're developing an application to shorten the dreaded "paper trail." One problem you have is that many of the forms found on these trails require people to sign their name—it's the accepted way to approve things. And sometimes the most important information may be hand drawn sketches, such as medical diagrams or traffic accident reports.

CIC InkTools,™ a software toolkit for Windows developers, lets you capture signatures, verify signatures, and manage hand drawn sketches–without burdening your programs with huge amounts of cumbersome graphics data. With InkTools you can capture and store signatures and sketches at high resolution in a very efficient compressed format. The electronic ink created with InkTools scales well and displays and prints at the highest possible resolution. This means your signature will really look like your signature and your handwritten notes will still be readable. InkTools helps your applications maintain the qualities that we treasure in paper forms while liberating your customers from paper document tracking nightmares.

InkTools comes with everything you need including 16 and 32-bit Windows DLLs and sample C code to help you get started. Add the power of ink to your applications.



| | Regular | O.T. | Vacation | Sick | Holiday | Total |
|---|---|---|---|---|---|---|
| Monday | 8.00 | | | | | 8.00 |
| Tuesday | 8.00 | 2.00 | | | | 10.00 |
| Wednesday | | | | | | 8.00 |
| Thursday | | | | | | |
| Friday | 8.00 | | | | | 12.00 |
| Saturday | | | | | | |
| Sunday | | | | | | |
| Total | 24.00 | 6.00 | 8.00 | | | 38.00 |

Signature Verified

Employee's Signature: [illegible]

**InkTools.™ Because sometimes the most important information isn't typed.**

CIC
**Communication Intelligence Corporation**
Redwood Shores, CA
*CIC is a public company Nasdaq symbol: CICI*

**Call 800.888.8CIC** DISCOUNT CODE:0110 **or visit us at www.cic.com**

CIRCLE NO. 305 ON READER SERVICE CARD

*(continued from page 62)*

the data, and *free* the i-node number for later use.

This scheme works well from a kernel point of view. The first smbfs implementation used exactly this mechanism. Then the first users complained that the command *pwd* could not find the current working directory. As the kernel did not complain about any inconsistent structure, the i-node numbering scheme seemed to be insufficient. The *getwd* routine, used by *pwd*, runs as follows: It works its way back to the root directory via the ".." entries of the directories. It stores the i-nodes of all the directories it finds on its way to the root. Then it reads all the directories as *ls* would, and compares the i-node numbers it gets with the ones found on the way to the root. Directories are read until the current directory is found. Not really efficient, but under UNIX it's the only way to find the path to the current directory.

It should now be clear why the numbering scheme I implemented has to fail for *getwd*: Each time a directory is opened for reading, a new i-node number is created for it. So *getwd* has no chance to reconstruct the path to the current directory. To help *getwd*, not only do the i-nodes used by the kernel need fixed and unique numbers, but so do all the i-nodes that are parents and grandparents of any i-node used, up to the filesystem's mount point. Thus ncpfs and smbfs build up a tree of i-nodes that represents the directory structure currently in use. When an i-node is released (because the corresponding file is closed or the directory has been left by *cd*, for example), the corresponding file-system structure is *free()*ed. Now the path to the root of the mount point is traversed and each directory's filesystem structure is released when it is not used by other parts of the directory tree.

By this enhanced scheme, *getwd* can be satisfied, but some as-yet-unfixed problems remain. In most cases, when you do a *ls – i* in an ncpfs-mounted directory, each directory entry is assigned the same i-node number. This is caused by the kernel *malloc* routine. When you *kfree()* a memory area and *kmalloc()* a memory block of the same size directly after, you get the same chunk of memory, resulting in identical i-node numbers for all directory entries. This is especially annoying because the nfs daemon relies on unique i-node numbers across the complete filesystem.

Another command that is broken by this scheme is an implementation of the *diff* command that tests whether the files have the same i-node, then refuses to compare the files if the numbers are the same. *cp* is another candidate: Sometimes it refuses to copy a file over an existing file, because it believes both to be the same file.

**Namespaces: NetWare has I-Nodes**

Again, Novell's NCP protocol was designed to implement DOS filesystem functionality. But when clients such as OS/2 and UNIX were used, NetWare had to cope with different notions of a filename. Although OS/2 and UNIX allow filenames that don't conform to the 8.3 convention, they do not agree about case sensitivity. To be able to cope with these needs, Novell introduced the concept of namespaces for NetWare 3. To handle namespaces, the NetWare core protocol was enhanced considerably.

From the protocol view, the NetWare 3 filesystem looks similar to UNIX filesystems. NetWare also seems to store i-nodes, which Novell calls "Directory Entries." The Directory Entry Table fulfills the same purpose as UNIX i-node tables. This central table contains everything relevant about a file, such as its size, its owner, and a reference to the file's data. The NCP namespace services let you access entries in this table and, thus, individual files by their indexes into this table. These indexes are 32 bits long and could be called the "i-node numbers" of the files.

There is one main directory entry per file, the original DOS filename entry. Each loaded namespace module adds another

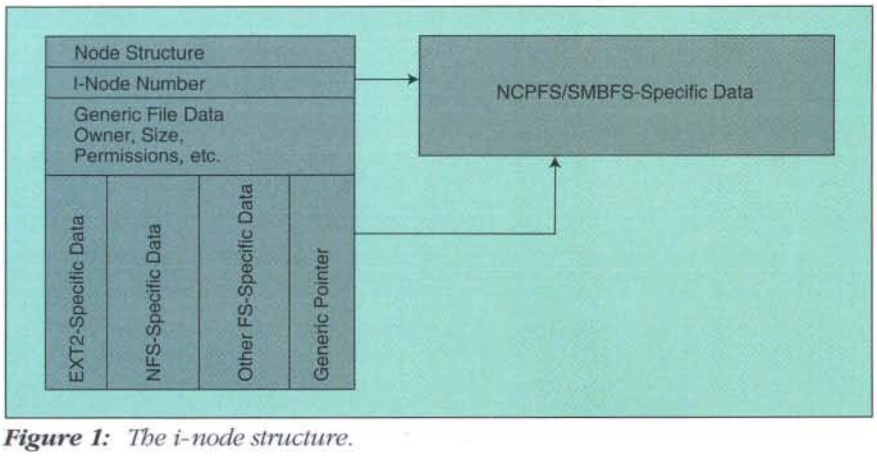


***Figure 1:*** *The i-node structure.*

**Install**Shield® Corporation

www.installshield.com

*(continued from page 64)*
entry for each file. This is a complete directory entry of its own and, as such, also has a unique 32-bit number. It contains the filename according to the conventions the namespace expects, along with the number of the DOS filename entry. For example, the NFS namespace module stores up to 255 bytes of the filename, and knows that uppercase and lowercase filenames are different.

When a file is created, it is always created in a special namespace. For example, Windows 95 creates all files in the OS/2 namespace, giving the file a long filename. This file has to be visible from all other namespaces, so the other entries have to be created by NetWare itself. NetWare squeezes the name into the 8.3 scheme required by DOS and creates the corresponding DOS directory entry. Windows 95 has to do the same for the VFAT filesystem, where each file has an 8.3 filename in addition to its long filename. When a DOS workstation creates a file, NetWare does not have to truncate the filename, but it still has to create the additional entries in all the namespaces loaded. For volumes with many small files, the way namespaces are used can cause problems because lots of additional directory entries are allocated.

You can look at the namespace information with the NDIR program that Novell delivers with NetWare. When you call NDIR with the option /L, it will display the long filenames that are assigned to the files. Windows 95 only shows you the OS/2 and DOS namespace filenames, but NDIR will also show you the NFS and other namespace's filenames.

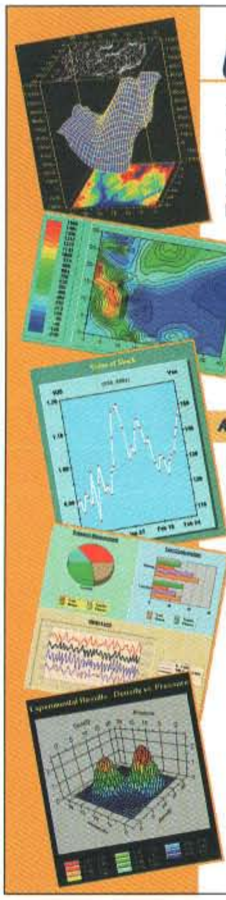
The namespace support of NetWare now makes it possible to solve the problem with the i-node numbers, as the protocol gives you 32-bit directory-entry numbers that uniquely identify files and directories in the NetWare filesystem. As in UNIX filesystems, where i-node numbers identify a file only within a filesystem, directory-entry numbers always refer to a NetWare filesystem, a volume. There might be two different files in different volumes with the same directory-entry number, as it is perfectly possible in UNIX. So to provide unique i-node numbers for ncpfs-mounted directories, you have to mount a single volume under one mount point. When you do this, the directory-entry numbers that NetWare shows the client via the NCP protocol are handed to user processes as i-node numbers, and user programs can rely on the i-node numbers to be unique on the complete filesystem. This way, the filesystems mounted can be reexported by nfs.

I implemented this as an option only, because it is possible that some NetWare servers have a lot of volumes, which would require many mount points for the complete server to be mounted. This could be expensive, since currently each ncpfs mount point uses one NetWare user license. The other issue is that standard Linux kernels allow only 64 mount points, which could be quickly exhausted. It's much cheaper to raise the Linux limit than the NetWare limit, but it's inconvenient. Consequently, you can still mount all volumes under one mount point, but you lose the reexportability.

## The Code

In general, the techniques I've described here are implemented in the ncpfs and smbfs kernel code, as well as parts of the general Linux vfs layer. More specifically, the files dir.c and inode.c (available electronically; see "Availability," page 3) are excerpts from the ncpfs code that illustrates the concepts I've presented. The latest versions of smbfs and ncpfs can always be found via http://www.kki.org/linux-lan/.

**DDJ**



**Ultimate Charting**

**Unprecedented performance and control – whether you develop for Windows or Motif, KL Group has the ultimate charting control for you. Loaded with unique features, Olectra Chart for Windows and XRT widgets for Motif are powerful enough to build virtually any 2D or 3D chart or graph:**

- X-Y Plots, Bar Charts, Pie Charts, Combination Graphs, Financial Graphs and Logarithmic Scientific Charts
- Real-time Updates to Build Scrolling Strip Charts
- Built-in Zooming, Scaling, Rotation
- Chart Labels and Annotations Anywhere
- Tunable Auto Precision Axis
- Dynamic Time & Date Axis on Any Plot

***Advanced technology that out-performs the competition***

**Windows:** With distinct 2D/3D OCXs and DLLs, Olectra Chart is more than 4X faster than First Impression 2.0 and Graphics Server SDK 4.0 in standard mode redraw speed. **Olectra Chart includes toll-free support and a money-back guarantee – at only $249!**

**Motif:** XRT/graph and XRT/3d are the world's leading Motif charting widgets. These award winning components can be used on over 18 UNIX platforms and can be easily integrated with all popular GUI builder tools. **Call today to get your free 30-day evaluation!**

KL GROUP

*The Leader in GUI Components*

**1-800-709-2947**

Tel: 416.594.1026 Fax: 416.594.1919
info@klg.com www.klg.com

* Prices in US dollars. Prices may be higher outside North America.
XRT/ is a registered trademark and Olectra Chart is a trademark of KL Group Inc.
All other product names are trademarks or registered trademarks of their respective owners. 007

**CIRCLE NO. 288 ON READER SERVICE CARD**

Available for Windows 3.1 and Windows 95.

VTOOLSD™

**SIMPLY THE BEST WAY TO WRITE VIRTUAL DEVICE DRIVERS FOR MICROSOFT WINDOWS.**

*Full sources included for:*

> Plug and Play hardware drivers
> Network driver templates
> VCOMM client driver
> VCOMM port driver outline
> File system drivers
> C Run Time Library
> C++ Class Library
> Dozens of examples

$495 FREE TECHNICAL SUPPORT No Royalties

***GUARANTEED TO SPEED VxD DEVELOPMENT!***

VtoolsD is a trademark of Vireo Software, Inc. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.

***www.vireo.com***

Vireo Software, Inc.
21 Half Moon Hill
Acton, MA 01720
Phone: 508-264-9200
Fax: 508-264-9205
Email: Sales@vireo.com

LEADTOOLS®

# The comprehensive imaging toolkit

ADTOOLS is a family of comprehensive imaging tools designed to help ogrammers integrate black & white, grayscale and color images into their plications as quickly as possible. LEADTOOLS is a collection of more than 200 nctions, properties and methods, with imaging technology in 10 general categories.

**anning**. Support for both 16 & 32 bit TWAIN interfaces for driving the most pular scanning devices. Developers have the option of using the default TWAIN terface of the datasource, or creating eir own. Additionally, both native and ffered RAM transfer modes are pported.

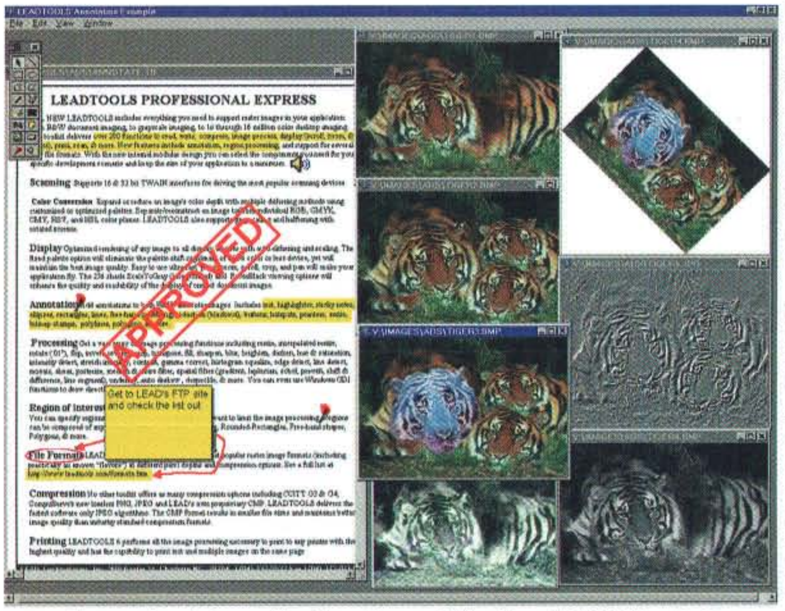
**olor Conversion**. Expand or reduce an age's color depth with multiple thering methods (8 available including oyd Stein & Burkes) using various lette options (including an auto-timized palette, or a user defined lette). Images can be separated and/or constructed to/from individual color anes, such as RGB & CMYK. ADTOOLS also supports grayscaling d halftoning with rotated screens.

**isplay**. Easy to use, ultra fast paint, om, scroll, crop, pan, auto-dithering d scaling will make the imaging rtion of the programmer's application . The 256 shade ScaleToGray nterpolated) and FavorBlack viewing tions will enhance the quality and adability of the display of 1-bit ocument images. Over 20 paint effects cluding wipes, blinds, spirals, various blocks, random pixels and lines.

**nnotations**. Annotations can be added to both black and white, and color images. elect from text, highlights, sticky notes, ellipses, free-hand scribbling, redaction lackout), hot-spots, audio, bitmap and text stamps, polygons and many more. Use w level or automated mode.

**nage Processing**. Over 50 functions, some only found in programs ch as PhotoShop. **Transforms** include resize, resample, rotate (.01 egree), flip, reverse, crop, shear, transpose, auto deskew, and combine tmap (with mathematical and Boolean operations). **Filters** include nderlay, fill, sharpen, blur, brighten, darken, invert, hue and saturation, tensity, contrast, gamma correction, histogram equalize, edge detect, ne detect, emboss, mosaic, posterize, median and noise filters, spatial lter (which can be pre-defined or customized), auto deskew, despeckle, nd more. **Drawing**. Programmers can access Microsoft Windows GDI functions uch as TextOut, BitBlt, Ellipse, and Rectangle), to draw directly to the bitmap's urface. **Region of Interest**. Process only a specific portion of an image (any ombination of rectangles, ellipses, rounded-rectangles, freehand shapes, polygons, transparent color and more) rather than the entire bitmap.

**Compression**. Includes JPEG, CCITT G3/G4, Huffman, Arithmatic, RLE, LZW, and LEAD's own proprietary CMP. LEADTOOLS Pro Express delivers the fastest software only JPEG algorithms available. The LEAD CMP formats results in smaller file sizes and maintains better image quality than industry standard compression formats.



*Screen capture showing a portion of the endless possibilities with new LEADTOOLS version 7*

**Image Format Import/Export filters**. Supports over 40 of the most popular raster image formats (including practically all known "flavors") at different pixel depths (1-32 bits) and compression options. The new version of LEADTOOLS adds Kodak's new FlashPix format, and several existing formats have been enhanced. (A separate license from Unisys is required to use the GIF and TIFF LZW code).

**Imaging for the Internet.** Specific features for the Internet. Support for Progressive JPEG, Progressive CMP, and GIF (interlace, transparency, animation, and embedded text). The GIF code has also been expanded to read and write the rare 2,3,5,6 and 7 bits per pixel formats. A FeedLoad function has been created to allow image data to be displayed as it is being transmitted across the net, providing Internet programmers with the fastest way to begin painting the client's screen. Examples for VB script and Java script are included.

**Imaging for Databases.** Specific features for the imaging database developer, such as Load/Save memory, Load/Save file offset, Visual Basic data binding, 32-bit ODBC, and a customized OLE 2.0 in-place server.

**Printing.** Performs all image processing necessary to print directly to any Windows supported printer, with the ability to print text and multiple images on the same page.

**Formats**

| | | | |
|---|---|---|---|
| BMP | IMG | MSP | TGA |
| CAL | IOCA | PCD | TIFF |
| DIB | JPEG | PCT | VDA |
| EPS | JFIF | PCX | VST |
| FAX (raw) | JTIF | PNG | WINFAX |
| DCX | LEAD CMP | PSD | WMF |
| GIF* | MAC | RAS | WPG |

***NEW! Progressive JPEG + CMP***

Partial list, full details at our new Web site.

LEADTOOLS saves programmers time by providing source code examples and sample applications, which can be directly added to applications. LEAD's mature code is backed up by a 30 day money-back guarantee (US & Canada only) and FREE technical support is available via phone, fax, BBS, Internet or CompuServe. For the most current and complete technical details, and comprehensive lists of supported file formats, image processing filters, functions, methods, and properties, please visit LEAD's web site at http://www.leadtools.com, or call (704) 332-5532.

(The ScaleToGray and FavorBlack functions, annotation code, and auto-deskew, despeckle, and region of interest features are only available in the Pro Express).

**CIRCLE NO. 90 ON READER SERVICE CARD**

**Download FREE imaging application built with NEW LEADTOOLS 7! or call**

**800-637-1837**

**Everything You Need To Know About Imaging**

**http://www.leadtools.com/**

LEAD TECHNOLOGIES INCORPORATED

900 Baxter St. Charlotte, NC 28204 704-332-5532 Fax: 704-372-8161 CompuServe: "GO LEADTECH"
LEADTOOLS is available in several versions, not all features are available in all versions.

| | | |
|---|---|---|
| WINDOWS 95 | WINDOWS NT | WIN32s |
| WINDOWS | OS/2 | DOS |
| VBX | OCX 16/32 | DLL 16/32 |

| | | |
|---|---|---|
| C/C++ | MFC | VISUAL BASIC |
| ACCESS | DELPHI | VISUAL FOXPRO |

* License required from Unisys for formats using LZW compression. LEAD and LEADTOOLS are registered trademarks of LEAD Technologies, Inc. All other product names are trademarks of their respective owners.

# Examining C++ Program Analyzers

*Finding out how programs really behave*

Scott Meyers and Martin Klaus

C++ has a well-deserved reputation for power and flexibility. It has an equally well-deserved reputation for complexity—its "gotchas" are legion. For example, omitting a virtual destructor in a base class typically leads to incomplete destruction of derived class objects when they are deleted through base-class pointers.

Experienced C++ programmers learn to avoid these kind of problematic constructs, but experience should not be necessary: Troublesome C++ can often be detected by static analysis, using tools that parse and analyze C++ source code. Such tools are becoming available, and during the summer and fall of 1996, we undertook an investigation to identify these tools and to assess their capabilities. In this article, we summarize the initial results of our investigation.

We were interested in answering three questions.

First, what tools statically analyze C++ programs and issue warnings about likely trouble spots? By focusing on static analysis, we limited our research to tools spiritually akin to lint. We explicitly ignored tools designed to detect dynamic (run-time) errors, such as programs that monitor memory usage and report on leaks. Such tools are important, but they offer functionality that complements—not replaces—static analysis. We also ignored tools that focus on lexical issues (identifier names, indentation style); our interest was in tools that identify constructs that affect program behavior.

Second, how comprehensive are the tools in identifying suspect C++ constructs? C++ has many facets, including data abstraction, inheritance, templates, and exception handling, and we wanted to find tools that checked for likely errors in many of these areas. A few tools checked only the C subset of C++; we ignored those offerings. Our interest was in tools for C++ programmers, and C++ programmers have different needs than C programmers.

Third, how well do the tools work on real programs? Can they parse real source code? Do they scale well when run on large projects? Are they robust enough to handle complex template instantiations, including those generated by the Standard Template Library?

In this article, we will address only the first two questions.

## Identifying Tools

When we began this project, we were aware of several static-analysis tools for C++, but we suspected there were others we didn't know about. Consequently, we posted a request for information to several USENET newsgroups, including groups devoted to C++ programming, OOP, and programming on various platforms. Based on the responses, we ultimately identified the tools discussed here.

- CodeCheck from Abraxas Software. CodeCheck is a stand-alone tool for DOS, Windows, and UNIX that lets you use a C-like language to specify what kinds of program analysis to perform. It comes with several predefined analysis programs, including some for computing program-complexity metrics and identifying non-portable code.
- C++Expert from CenterLine Software. Also a stand-alone tool, C++Expert performs static and dynamic analyses of C and C++ programs. Its static checks are drawn from Scott Meyers' *Effective C++* and *More Effective C++*, and its diagnostics contain hypertext links to online versions of those books. At this time, it supports only UNIX.
- FlexeLint/PC-Lint from Gimpel Software. Another stand-alone tool (the name is FlexeLint for UNIX, and PC-Lint for DOS, Windows, and OS/2), FlexeLint/PC-Lint is perhaps truest to the classic lint tradition. It can check for over 600 potential error conditions in C and C++ source code, including conditions that affect more than one translation unit or that require detailed dataflow analysis.
- CodeAdvisor from Hewlett-Packard. CodeAdvisor is a part of HP's SoftBench development environment for UNIX. It enforces 23 predefined rules, and you can extend its capabilities by coding new analyses in C++, then linking them in. Source-code information is stored in a database, so it is possible to perform checks that involve multiple translation units.
- CodeWizard from ParaSoft. CodeWizard is a stand-alone UNIX tool designed to enforce a set of 24 rules selected from *Effective C++*.
- QA/C++ from Programming Research. Another stand-alone tool for UNIX, QA/C++ works in two phases. First, it examines C or C++ source code and stores the results in a database. Different Programming Research analysis tools may then be run against the

*Scott, a software-development consultant and author of* Effective C++ *and* More Effective C++, *can be contacted at smeyers@netcom.com. Martin holds a degree in computer science from the Johannes Kepler University, in Linz, Austria, and can be contacted at mklaus@swe.uni-linz.ac.at.*

NEW RELEASE!

From the *Dr. Dobb's CD-ROM Library,*

## Essential Books on File Formats CD-ROM

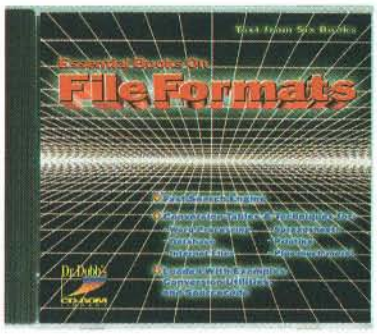
Look no further!

The *Essential Books on File Formats CD-ROM* contains the complete text from six books, providing the most comprehensive and detailed collection of information on all the important and popular file formats in use today.

Selected by the editors of *Dr. Dobb's,* this CD-ROM contains invaluable information on file formats used for graphics, multimedia, sound, databases, spreadsheets, Windows, the Internet, and much more!

No matter what your programming focus, the *Essential Books on File Formats CD-ROM* discloses the secrets and insider knowledge you need to make your programming instantly compatible with all the major applications — right from the start.

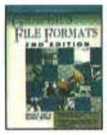
Plus, the CD-ROM's powerful full-text search engine and hyperlink capabilities allow you to search quickly and easily across all the books and link directly to the information you need.



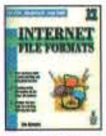
Includes these Six Essential Books selected by the *Dr. Dobb's* editorial staff

***The File Formats Handbook***
by Günter Born

***More File Formats for Popular PC Software***
by Jeff Walden

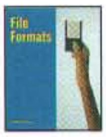


***Graphics File Formats, 2nd Edition***
by David C. Kay & John R. Levine



***Internet File Formats***
by Tim Kientzle

***Inside Windows File Formats***
by Tom Swan



***File Formats***
by Allen G. Taylor

## Order Today!

U.S. & Canada: **800-992-0549**
All Other Countries: **913-841-1631**
E-mail: **orders@mfi.com**
Fax Orders: **913-841-2624**

Mail Orders:
Dr. Dobb's CD-ROM Library
1601 W. 23rd St., Ste. 200
Lawrence, KS 66046-2700 USA

**Only! $69.95**

## Order Today!

Please send my *Essential Books on File Formats CD-ROM* today!
My price is $69.95 (plus shipping: $2.00 U.S./Canada; all other countries $10.00)
Please add sales tax in the following states: CA (8.5%), GA (6%), IL (6.25%), KS (6.9%, NY (8.25%), TX (8.25%)

Name

Address

City State ZIP Country

Home Phone Work Phone

Email Address ☐ Check ☐ VISA ☐ Mastercard ☐ Amex

Credti Card#: exp.date

Signature 97DJ

**Visit our web site for more information! www.ddj.com/cdrom/**

*(continued from page 68)*

database; these tools generate warning messages. There is no database API that lets programmers develop their own analyses.

- The Apex C/C++ Development Environment from Rational Software. Among other capabilities, the Apex environment enforces 22 predefined rules for C and C++ programming under UNIX.

To these choices, we added our noncommercial program, CCEL, purely for purposes of comparison. CCEL began as a research project on static analysis of C++ programs under the direction of one of us (Meyers) and was eventually fully implemented through independent work by the other (Klaus). We added CCEL to our investigation because we were familiar with its capabilities and limitations, and we felt it would be interesting to compare commercial approaches to our research-based initiative.

## Our Approach

There were three phases in our testing process.

1. We developed a set of benchmark rules constraining the structure of C++ programs. For example, one rule is that all base classes must have virtual destructors. We tried to develop a set of rules that was representative of the kinds of rules that real programmers would find useful.

2. We contacted vendors and asked which rules their tool could enforce. This information proved useful during our empirical tests, because discrepancies between vendor claims and our findings often identified subtle differences between our rules and those enforced by vendors.

3. We developed of a set of sample source files seeded with rule violations. We ran each tool on each source file to see whether the seeded rule violation was correctly identified.

Our results yielded Table 2, which shows how well each tool enforced our benchmark rules on our benchmark programs.

## Choosing Rules

There are many ways to compose a set of benchmark rules for C++ programs, but it is difficult to argue that one set is "better" than another. As a result, we made no attempt to develop the "best" set of rules. Instead, we fell back on the fact that one of us (Meyers) has authored two books containing guidelines for C++ programming and we chose nearly all our rules from those books.

This approach is not as gratuitous as it might appear. Meyers' *Effective C++* and *More Effective C++* have been well-received in the C++ programming community, and one or both form the basis for many sets of corporate-coding guidelines. In addition, these books form the basis for at least two of the static-analysis tools in our investigation. Finally, by drawing our rules from well-known and

| Rule | Book | Item | Description |
|---|---|---|---|
| (a) | | | |
| 1 | E | 1 | Use *const* instead of *#define* for constants at global and file scope. |
| 2 | M | 2 | Use *new*-style casts instead of C-style casts. |
| 3 | M | 3 | Don't treat a pointer to *Derived[]* as a pointer to *Base[]*. |
| (b) | | | |
| 4 | E | 5 | Use the same form for calls to *new* and *delete*. (In general, this calls for dynamic analysis, but static analysis can catch some special cases; calls to *new* in constructors and to *delete* in destructors, for example.) |
| 5 | E | 6 | When the result of a *new* expression in a constructor is stored in a dumb pointer class member, make sure *delete* is called on that member in the destructor. |
| 6 | E | 9 | Avoid hiding the default signature for operator *new* and operator *delete*. |
| (c) | | | |
| 7a | E | 11 | Declare a copy constructor for each class declaring a pointer data member. |
| 7b | E | 1 | Declare an assignment operator for each class declaring a pointer data member. |
| 8 | E | 12 | Initialize each class data member via the member initialization list. |
| 9 | E | 13 | List members in a member initialization list in an order consistent with the order in which they are actually initialized. |
| 10 | E | 14 | Make destructors virtual in base classes. |
| 11 | E | 15 | Have the definition of *operator=* return a reference to **this*. (Note: This says nothing about declarations.) |
| 12a | E | 16 | Assign to every local data member inside *operator=*. |
| 12b | E | 16 | Call a base class *operator=* from a derived class *operator=*. |
| 12c | | | Use the member initialization list to ensure that a base class copy constructor is called from a derived class copy constructor. |
| 13 | | | Don't call virtual functions in constructors or destructors. |
| (d) | | | |
| 14 | E | 19 | Use nonmember functions for binary operations like +–/* when a class has a converting constructor. |
| 15 | E | 20 | Avoid public data members. |
| 16 | E | 22 | Use pass-by-*ref*-to-*const* instead of pass-by-value where both are valid and the former is likely to be |
| 17a | E | 23 | Have operators like +–/* return an object, not a reference. |
| 17b | M | 6 | And make those return values *const*. |
| 18 | E | 25 | Don't overload on a pointer and an *int*. |
| 19 | M | 33 | Make nonleaf classes abstract. |
| 20 | M | 24 | Avoid gratuitous use of virtual inheritance; that is, make sure there are at least two inheritance paths to each virtual base class. |
| (e) | | | |
| 21 | E<br>E | 29<br>30 | Don't return pointers/references to internal data structures unless they are pointers/references to *const*. |
| 22 | M | 26 | Never define a static variable inside a nonmember inline function unless the function is declared *extern*. (Note: In July 1996, changes to the nascent standard for ANSI/ISO C++ obviated the need for this rule, at least on paper. However, the need still exists in practice, because many compilers continue to heed the older rules that can lead to duplicated variables in inline nonmember functions.) |
| 23 | | | Avoid use of "..." in function parameter lists. |
| (f) | | | |
| 24 | | 3 | Don't redefine an inherited nonvirtual function. |
| 25 | E | 38 | Don't redefine an inherited default parameter value. |
| (g) | | | |
| 26 | M | 5 | Avoid use of user-defined conversion operators (nonexplicit single-argument constructors and implicit type conversion operators). |
| 27 | M | 7 | Don't overload &&, \|\|, or ,. |
| 28 | M | 6 | Make sure operators ++ and –– have the correct return type. |
| 29 | M | 6 | Use prefix ++ and –– when the result of the increment or decrement expression is unused. |
| 30 | M | 22 | Declare *op=* if you declare binary operator (for example, declare += if you declare +, declare –= if you declare –). One way to satisfy this constraint is by providing a template that yields the appropriate function. |
| (h) | | | |
| 31 | M | 11 | Prevent exceptions from leaving destructors. |
| 32 | M | 13 | Catch exceptions by reference. |

***Table 1:** Benchmark rules. (a) General; (b) use of* new *and* delete*; (c) constructors/destructors/assignment; (d) design; (e) implementation; (f) inheritance; (g) operators; (h) exceptions.*

easily accessible sources, we avoided the need to explicitly justify individual rules in our benchmark set. Instead, the justification for nearly every rule is available in the books, and we simply refer to the appropriate book location as the rationale for each rule.

We chose 36 rules divided into eight categories; see Table 1. Each rule begins with its "Rule" number, followed by a reference to either *Effective C++* (E) or *More Effective C++* (M). Next is a reference to the book "Item" number from which the rule is derived. The text of the rule is often different from the text of the book Item, because the book Items tend to be worded too generally to be checked.

Some of the rules may seem controversial, especially in light of the C++ found in many popular class libraries. Rule 15 (no public data members) is widely violated in the MFC, for example, while almost no library adheres to Rule 19 (make all nonleaf classes abstract). With the exceptions of Rules 13 and 23 (which we hope are self explanatory), *Effective C++* and *More Effective C++* offer firm technical foundations for each rule. We believe it is therefore important that programmers be able to enforce those constraints, even if the majority of programmers choose not to. Furthermore, our decision to include rules that are commonly violated helps us evaluate the effectiveness of the tools' filtering capabilities. (We do not report on this aspect of the tools in this article, but it is an important consideration in the practical application of any tool.)

## Benchmark Programs

For each of our 36 rules, we developed a source file seeded with a violation of the rule. We then executed each tool on each source file to see if the tools correctly identified the seeded errors. These source files were truly trivial— many were under ten lines long. Our goal was not to provide a realistic test of the tools— just to see whether or not the tools could identify rule violations in the simplest of cases. (Sometimes, this backfired and yielded misleading results.) Listing One (listings begin on page 87) is the source code for the file used to test Rule 20.

## Compilers versus Special Tools

Several people responded to our request for information on static-analysis tools by remarking that they found little need for such tools. Instead, they relied on compilers to flag conditions that were likely to lead to trouble ("I find GNU G++ with –ansi –pedantic –Wall –O flags useful," was a typical comment).

In fact, the GNU compiler was singled out as being especially good at warning about troublesome C++. This piqued our curiosity about compiler warnings. How many of our candidate rules would compilers identify?

To find out, we submitted our benchmark programs to five compilers, in each case enabling as many warnings as possible. As Table 2 shows, the results were disappointing. Even G++ identified, at most, 2 of the 36 rule violations, and three of the compilers identified none. This confirmed our impression (based on our experience as C++ programmers) that while compilers— at least the compilers with which we have had experience— are good at many things, identifying legal, but potentially troublesome, C++ source code is not one of them.

## Specifying Constraints

The tools in our study let you specify what conditions to check for in one of two ways. Most tools follow the lint model, whereby the tool is created with the ability to enforce some set of predefined constraints, and you turn these constraints on or off. There is no way to extend the capabilities of such tools. For example, a tool is either capable of detecting that an exception may leave a destructor (Rule 31) or it's not. If it's not, there is no way for a tool user to add that capability.

A different approach— employed by Abraxas' CodeCheck, HP's CodeAdvisor, and our CCEL— is to provide tool users with a language in which to express constraints of their own. Such tools may or not be useful "out of the box" (it depends on the existence and utility of predefined rule libraries), but can be extended to check for new, user-defined conditions. This approach is more powerful, but, as in the case of C++ itself, complexity often accompanies power; the power is inaccessible until you have mastered the constraint-expression language. Furthermore, the addition of user-defined constraints may affect an analysis tool's performance, because enforcement of such constraints may require arbitrary amounts of time, memory, or other resources.

We made no attempt to master the various constraint-expression languages used by the different tools, but the examples we saw (see the accompanying text box entitled "Constraint Expression Languages") reinforced the lessons we learned during the design and implementation of CCEL— it's hard to design a language for expressing constraints on a language as feature-filled as C++, and such a constraint language is nontrivial to learn. Abraxas, for example, reports that it takes between three and six months to become proficient in the CodeCheck constraint language. Most Abraxas customers want to hire specialists to compose rules instead of having to learn to write the rules themselves.

Most programmable tools attempt to offer the best of both worlds by shipping a set of predefined rule libraries that check for commonly desired constraints. This eliminates the need to write rules to cover common constraints.

## Results and Discussion

Table 2 presents the results of running the various tools on the collection of benchmark programs. Several features are of interest. First, no tool was able to enforce all of our 36 benchmark rules, not even the tools supporting user-defined constraints. Thus, even the best of tools currently available offers only partial coverage of C++. This is especially noteworthy because our benchmark rules themselves failed to exercise all major language features; templates are a particularly obvious omission.

Second, the number of benchmark rules that can be enforced without programming (out of the box) is, at most, 17 of 36. (CCEL supports 19, but CCEL is a research project, not a commercial tool.) If we speculate that our set of benchmark rules is somehow representative of the kinds of constraints real programmers might want to enforce, this suggests that current tools cover, at best, only about half of those constraints. Of course, automatic enforcement of half a set of requirements is better than no enforcement at all, but the data in Table 2 suggest that there is much room for increased language coverage by static-analysis tools for C++.

Third, it is not uncommon to have subtle mismatches between a benchmark rule and the conditions detected by the analysis tools. In most cases, this is an outgrowth of the vendors' attempts to avoid generating warning messages when no truly harmful condition exists. For example, consider Rule 10: "Make destructors virtual in base classes." Many programmers consider this rule too aggressive, and a common alternative form of the same rule is: "Make destructors virtual in classes containing virtual functions." This form has the advantage that no virtual table pointer is added to a class simply to satisfy the rule. (This is the rule variant that's employed by the GNU C++ compiler, HP's CodeAdvisor, and Programming Research's QA/C++.)

The motivation for this rule (in any form) is that Listing Two is generally harmful if the base class lacks a virtual destructor. In truth, Listing Two is only harmful if one or more of the following conditions holds:

- D has a destructor.
- D has data members that have destructors.

- D has data members that contain data members (that contain data members, and so on) with destructors.

At least one tool vendor attempts to issue a diagnostic only if these more stringent conditions exist, and the conditions do not exist in our test program (Listing Three). The tool in question thus issues no diagnostic on our sample program, but if class *Derived* were nontrivial, the tool might issue a warning.

This more precise analysis should be beneficial for users, because a diagnostic should be issued only if a problem truly exists. However, the rules of C++ can be both complicated and unintuitive, and their subtlety can cut both ways. In the case of the vendor attempting to check for the more detailed conditions outlined earlier, the test for data members with destructors in the derived class was omitted. Hence, though the tool avoids issuing warnings in harmless cases, it also avoids issuing warnings in some harmful, but rare cases. These are precisely the cases in which static-analysis tools that correctly understand the detailed rules of C++ are most useful!

Another tool had trouble issuing correct diagnostics when compiler-generated functions— default constructors, copy constructors, assignment operators, and destructors (especially derived-class destructors)—were involved. Because of the minimalist nature of our test cases, our programs had many instances of such functions; this led to incorrect results from some tools.

Whether such shortcomings would cause problems when the tools are applied to real programs is unknown, but it hints at a deeper problem we found: Vendors don't seem to understand the subtleties of C++ as well as they should. We believe that vendors of C++ analysis tools must understand C++ as well as compiler vendors, but based on our experience with the tools in this study, we must report that such expertise cannot yet be taken for granted.

**Caveats**

While Table 2 provides insight into the state of existing lint-like tools for C++, it is important to recognize what it does not show. We were interested only in the capability of such tools to handle the "++" part of C++, but most of the tools also provide significant other capabilities.

Most tools also check the "C" part of C++, some quite extensively. This can be useful. By limiting our tests specifically to C++ capabilities, we were able to sharpen our focus, but we also screened out the majority of some tools' functionality.

Many tools offer stylistic and lexical checks in addition to the semantic issues we looked at. For example, if you wish to ensure that classes never use the default access level of *private,* but instead declare it explicitly, at least one tool will note violations of that constraint.

Some tools offer complementary analyses in addition to checking coding "style." For example, Programming Research's QA/C++ can calculate various program-complexity metrics.

In addition, our set of benchmark rules was far from exhaustive. Some vendors check for C++-specific conditions we didn't consider; Table 2 says nothing about such capabilities.

All this is to say that Table 2 is anything but a buyer's guide. Furthermore, there are many nontechnical characteristics of analysis tools you should consider before deciding which, if any, is suitable for your circumstances. The following questions come to mind:

- How easy is it to install, configure, and use the tool? These factors are especially important for tools with equivalent (or close to equivalent) capabilities. For example, Gimpel Software's FlexeLint and Productivity Through Software's ProLint use the same underlying analysis engine, but offer quite different user interfaces.
- How easy is it to filter out unwanted diagnostics? The traditional Achilles Heel of lint-like tools is an unacceptable signal-to-noise ratio, so it's important that users be given fine-grained control over what code is analyzed and which diagnostics appear. In fact, some vendors deliberately avoided offering checks for some conditions (for example, the use of preprocessor macros to define constants— our Rule 1) because they felt it would be more bothersome than useful to their customers. (Respondents to our newsgroup postings indicated that a bad signal-to-noise ratio is a continuing problem, even with some of the tools considered here.)
- How robust and up-to-date is the C++ parser? To be maximally useful, a C++ analyzer must parse exactly the same language as the compiler(s) you use. It's particularly frustrating if the analyzer rejects code your compiler accepts.

## Constraint Expression Languages

As an example of the different ways in which constraints on C++ programs may be expressed, consider our Rule 10. In English, the constraint is: "Make destructors virtual in base classes." When this constraint is formalized and made amenable to enforcement by computer, it quickly becomes more complicated.

CCEL offers a fairly succinct way to express this constraint, because CCEL was specifically designed to make the expression of constraints like this straightforward. Still, CCEL's formal nature makes it wordier than English. (It also makes it more precise.) Example 1 illustrates Rule 10 in CCEL.

CCEL embodies a declarative approach to the specification of constraints: You specify what you want to enforce, not how to enforce it. This CCEL constraint can be read like this: "For all classes B and all classes D that inherit from B, B must have a member function *m* such that *m* is a destructor and *m* is virtual."

Compared to CCEL, the programmable analysis tools we investigated are long winded. That's because they employ a procedural approach to constraint specification: You write code specifying how to go about detecting violations of the constraints you define. In practice, a procedural approach is more powerful, but it's also more complicated.

For example, Abraxas offers a C-like programming language for new constraints. HP's CodeAdvisor uses C++ as an extension language and new constraints are implemented via classes inheriting from the predefined *Rule* class. These classes are then compiled and linked to a run-time library. Centerline plans a similar extension of C++Expert.

Implementations of Rule 10 in all of these tools are available electronically (see "Availability," page 3).

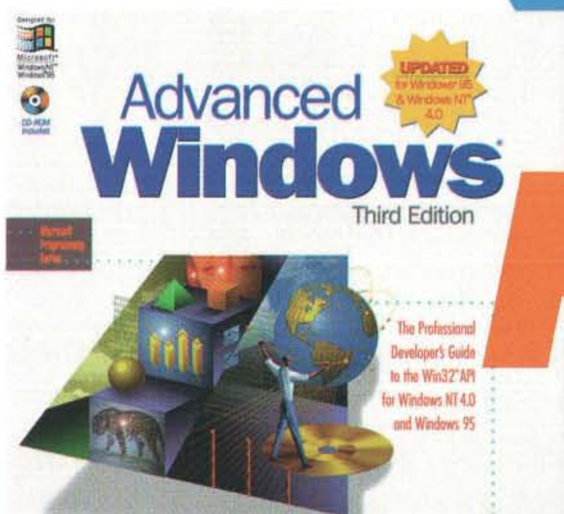
—S.M. and M.K.

```
BaseClassDtor (
  Class B;
  Class D ¦ D.is_descendant(B):
   Assert(MemberFunction B::m; ¦ m.name() == "~" +  B.name() && m.is_virtual());
);
```

***Example 1:*** *Rule 10 in CCEL.*

Win32

Unleash the Power


Advanced
Windows
Third Edition
UPDATED for Windows 95 & Windows NT 4.0
The Professional Developer's Guide to the Win32 API for Windows NT 4.0 and Windows 95
Jeffrey Richter
Microsoft Press

Advanced Windows®, Third Edition
1-57231-548-2 $49.99 ($67.99 Canada)

**The preeminent source of information on programming for Win32®, ADVANCED WINDOWS, Third Edition, unveils important recent enhancements, including support for Windows NT® 4.0. To unleash the powerful capabilities of the 32-bit API, and to create state-of-the-art programs for the Windows 95 and Windows NT operating systems, start with ADVANCED WINDOWS, Third Edition, from your favorite bookstore or software store.**

***Microsoft*® *Press***

**Available in quality bookstores and computer stores worldwide.**
To locate your nearest source for Microsoft Press® products, reach us at 1-800-MSPRESS in the U.S., or ***www.microsoft.com/mspress/***

Designed for Microsoft® Windows NT® Windows®95

CIRCLE NO. 319 ON READER SERVICE CARD

| | Compilers | | | | | Static-Analysis Tools | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Product** | **Borland C++** | **Gnu g++** | **Visual C++** | **SunSoft C++** | **Symantec C++** | **CodeWizard** | **PC-Lint** | **CodeCheck** | **QA/C++** | **CCEL** | **CodeAdvisor** | **C++Expert** | **Apex** |
| **Version** | 5.0 | 2.7.2 | 10.00.5270 | SC3.0.1 | 7.20B1n | 1.0 | 7.00c | 6.04 B3 | 3.0.1 | N/A | C.05.00 | 1.0 | 2.0.6C |
| **Options** | -w | -ansi -Wall | -W4 | +w2 | -w- | unsuppress all | -w4 | Note 14 | (Defaults) | N/A | (Defaults) | Not Tested | (Defaults) |
| **Tested Under** | NT 3.51 | SunOS 4.1.4 | NT 3.51 | Solaris 2.5 | NT 3.51 | Solaris 2.5 | NT 3.51 | NT 3.51 | Solaris 2.5 | Solaris 2.5 | Solaris 2.5 | Not Tested | Solaris 2.5 |
| **1** | - | - | - | - | - | - | - | X | X | X | CWP | - | - |
| **2** | - | - | - | - | - | - | - | CWP | X | - | - | - | - |
| **3** | - | - | - | - | - | - | Note 5 | CWP | - | - | CWP | CD | - |
| **4** | - | - | - | - | - | Note 4 | - | CWP | Note 4 | - | CWP | CD | - |
| **5** | - | - | - | - | - | X | X | CWP | - | - | CWP | CD | - |
| **6** | - | - | - | - | - | X | - | CWP | - | X | CWP | X | Note 3 |
| **7a** | - | - | - | - | - | X | Note 6 | CWP | X | X | Note 6 | X | - |
| **7b** | - | - | - | - | - | X | Note 6 | CWP | X | X | Note 6 | X | - |
| **8** | - | - | - | - | - | Note 3 | X | Note 11 | X | - | CWP | Note 13 | - |
| **9** | - | X | - | - | - | X | X | Note 11 | X | - | Note 7 | X | - |
| **10** | - | Note 2 | - | - | - | Note 3 | Note 10 | Note 2 | Note 2 | X | Note 2 | Note 3 | X |
| **11** | - | - | - | - | - | Note 3 | - | CWP | Note 3 | - | CWP | X | - |
| **12a** | - | - | - | - | - | X | - | CWP | - | - | CWP | - | - |
| **12b** | - | - | - | - | - | - | - | CWP | - | - | CWP | - | - |
| **12c** | - | - | - | - | - | - | Note 7 | Note 11 | X | - | CWP | - | - |
| **13** | - | - | - | - | - | - | X | CWP | - | - | X | - | - |
| **14** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **15** | - | - | - | - | - | X | - | CWP | X | X | CWP | X | X |
| **16** | - | - | - | - | - | X | - | CWP | Note 3 | X | CWP | Note 3 | - |
| **17a** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **17b** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **18** | - | - | - | - | - | Note 3 | - | CWP | - | X | CWP | X | - |
| **19** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **20** | - | - | - | - | - | - | - | - | - | X | CWP | - | - |
| **21** | - | - | - | - | - | X | Note 3 | CWP | Note 3 | - | CWP | - | - |
| **22** | Note 1 | - | - | - | - | - | - | CWP | Note 7 | - | CWP | - | - |
| **23** | - | - | - | - | - | - | - | X | X | X | CWP | - | - |
| **24** | - | - | - | - | - | X | X | CWP | - | X | Note 3 | X | - |
| **25** | - | - | - | - | - | X | - | - | - | X | CWP | X | - |
| **26** | - | - | - | - | - | - | - | CWP | Note 3 | - | Note 12 | - | - |
| **27** | - | - | - | - | - | - | - | CWP | - | X | CWP | X | - |
| **28** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **29** | - | - | - | - | - | - | - | CWP | X | - | - | - | - |
| **30** | - | - | - | - | - | - | - | CWP | - | X | CWP | - | - |
| **31** | - | - | - | - | - | - | - | CWP | - | - | - | - | - |
| **32** | - | - | - | - | - | - | - | CWP | - | - | - | X | - |

| | |
|---|---|
| **Note 1** | Warning issued that functions containing static variables are not expanded inline. |
| **Note 2** | Will note a nonvirtual destructor in classes with virtual functions. |
| **Note 3** | Vendor claims the condition will be detected, but tests exposed at least one failure to do so. |
| **Note 4** | A diagnostic is issued only when the calls to *new* and *delete* are within the same function. |
| **Note 5** | Erroroneously diagnosed an error in a valid source file. |
| **Note 6** | Doesn't diagnose this particular condition, but uses similar heuristics to identify classes where assignment operators and copy constructors should be declared. |
| **Note 7** | A diagnostic is issued that hints at the problem, but the problem is not directly identified. |
| **Note 8** | A diagnostic is issued only when the member is "initialized" in the body of the constructor. |
| **Note 9** | (This note is not used.) |
| **Note 10** | Attempts to warn about this only when the condition would cause a run-time problem, but tests showed at least one failure to do so. |
| **Note 11** | The vendor claims that users can program the tool to detect violations of the constraint, but technical support was unable to describe how when asked. |
| **Note 12** | A constraint to enforce the rule can be written, but the HP parser doesn't yet support "explicit." |
| **Note 13** | This condition isn't detected, but use of uninitialized memory is dynamically detected. |
| **Note 14** | *-Rcplus* used, but Abraxas describes this as a "tutorial and example file" not designed for production use. |

***Table 2:** Results of submitting sample programs to tools: –, tool failed to detect violations of this rule on our benchmark programs; X, tool detected the seeded rule violation in our benchmark programs; C, vendor claims tool will detect violations of this rule, but we were unable to acquire a copy of the tool to test claim; CD, vendor claims tool will detect violations of this rule dynamically, but we were unable to acquire a copy of the tool to test claim; CWP, vendor claims users can program the tool to detect violations of the constraint. We did not test the claim.*

*(continued from page 72)*

- Can the tool handle large projects—those with multiple libraries and hundreds or thousands of source files? Based on the responses we got from our USENET postings, the answer too often is that it cannot.
- Is the documentation complete, accurate, accessible, and comprehensible?
- Does the vendor offer adequate customer service, including technical support?
- How well established is the vendor? Is the vendor likely to continue to support the tool for years to come?

Our study considered none of these issues.

Finally, it is important to remember that Table 2 is based on tests we performed in August/September 1996. Virtually all of the tools we examined are under active development, so it's likely that new versions exist even as you read this report. For example, we know that Abraxas is currently beta-testing a set of predefined constraints derived from material in Meyers' books, and CenterLine and Rational are planning upgrades to C++Expert and Apex, respectively, that will allow users to define new constraints. Other vendors are similarly active. Table 2 represents a mere snapshot of the commercial state of the art in September 1996.

## Summary

A number of analysis tools are now available that read C++ source code and warn about possible behavioral problems. They cover varying aspects of C++, though none offers truly comprehensive coverage of the language. Based on simple tests, we believe that many dangerous C++ constructs can be detected, though the complexity of C++ leads to incorrect behavior on the part of some tools, especially where compiler-generated functions are concerned. C++ analysis tools are under active development, and it is likely that the data in this article fails to accurately reflect the current capabilities of the tools we examined. If you are interested in static-analysis tools for C++, we encourage you to contact the vendors, conduct your own tests, come to your own conclusions—then share them with us.

## Acknowledgment

We are grateful to Jill Huchital for her comments on a draft of this article.

## References

Meyers, Scott. *Effective C++*, Reading, MA: Addison-Wesley, 1992.

———*More Effective C++*, Reading, MA: Addison-Wesley, 1996.

Meyers, Scott, Carolyn K. Duby, and Steven P. Reiss. "Constraining the Structure and Style of Object-Oriented Programs." *Principles and Practice of Constraint Programming*. Cambridge, MA: MIT Press, 1995.

Musser, David R. and Atul Saini. *STL Tutorial and Reference Guide*, Reading, MA: Addison-Wesley, 1996.

**DDJ**

**(Listings begin on page 87.)**

### For More Information

Abraxas Software
5530 SW Kelly Avenue
Portland, OR 97201
503-244-5253
http://www.abxsoft.com/

Centerline Software
10 Fawcett Street
Cambridge, MA 02138-1110
617-498-3000
http://www.centerline.com/

Gimpel Software
3207 Hogarth Lane
Collegeville, PA 19426
610-584-4261
http://www.gimpel.com/

Hewlett-Packard
19410 Homestead Road
Cupertino, CA 95014-0604
408-725-8900
http://www.hp.com/sesd/CA/

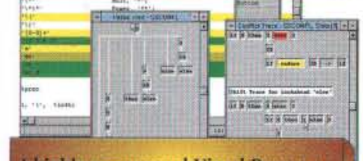
ParaSoft Corp.
2031 South Myrtle Avenue
Monrovia, CA 91016
818-305-0041
http://www.parasoft.com/

Productivity Through Software Inc.
555 Bryant, Suite 555
Palo Alto, CA 94301
415-934-3200
http://www.pts.co.uk/

Programming Research Ltd.
1/11 Molesey Road, Hersham
Surrey KT12 4RH, UK
+44-1932-88 80 80
http://www.prqa.co.uk/

Rational Software Corp.
2800 San Tomas Expressway
Santa Clara, CA 95051-0951
408-496-3600
http://www.rational.com/

**Need a Modern Parsing Tool??**



I highly recommend Visual Parse++
Larry O'Brien, Software Development

**Visual Parse++** is the only tool that applies the techniques of visual programming and RAD (Rapid Application Development) to lexing and parsing technology. Why settle for a text-based tool when you can:

† See the Syntax Tree
† See the Parsing Stack
† See Conflict Trace Trees
† See Grammar Rule Matches
† See Errors and Error Recovery
† See Regular Expression Matches
† See the Flow of your Input Data
**† Work in one tenth the time !**

When your design is finished, write your application using:

**C/C++**: full support for VC++ 1.5, 2.x, 4.x and BC++ 4.x (16 and 32 bit). Comes with a complete, 100% portable class library (Unix, Mac, OS/2)

**Visual Basic**: use the powerful parsing VBX/OCX

**Delphi**: native Delphi support!

Comes with drop-in parsers for **HTML, SQL, RTF**, Modula 2, C, C++, CONFIG.SYS, and more.

Big claims? We back them up with a 60-day money back guarantee. If it doesn't do what we say, send it back!

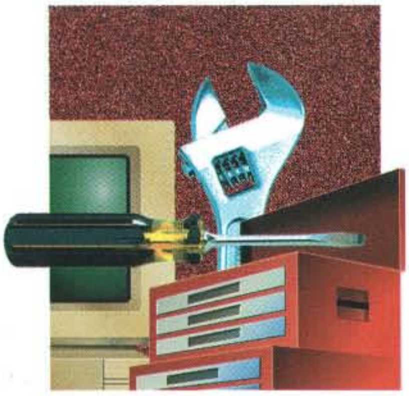
**Intro price: $299!**

**Call (800) 988-9023**

SandStone technology

950 Shore Crest Road
Carlsbad, CA 92009
Phone: (619) 929-9778
Fax: (619) 929-9848

**CIRCLE NO. 203 ON READER SERVICE CARD**



# Testing Testers

*Selecting the right tool for the right job*

Ron van der Wal

In an ideal world, software development would progress smoothly from requirements to completion. In the real world, however, errors creep in. To get them out, various automated checking tools are available, including BoundsChecker, CodeGuard, Purify, PC-lint, and others. Basically, these tools watch your program in operation and report on errors. (The exception is PC-lint, which operates on your source code.) Which checking tool is best for you? Every vendor claims its tool is the best—and they all have the numbers to prove it. So what are you to do?

To answer this question, I worked with a number of tools in my development efforts, using a number of tests—some from vendors, some of my own (mine are available electronically; see "Availability," page 3)—to compare the tools. What I discovered (not surprisingly) is that each tool has strengths and weaknesses. No single checking tool is ideal in all respects, but neither are there any bad tools among the crop that I examined—it just depends on what you're looking for.

The testing tools I examine here fall into two categories—static and dynamic. Static testers do their job without executing the program under test, and dynamic testers monitor its execution (for more information, see the accompanying text box entitled "Static and Dynamic Testing"). PC-lint is the only static tester in the roundup; all others are dynamic testers. Dynamic testers use several different methods to monitor the program, but one way or another, they all insert some sort of probe into your program. These probes are known as "instrumentation," and many differences between the tools can be traced back to their instrumentation method (see accompanying text box entitled "Instrumentation Techniques").

*Ron, author of the Tarma Simulation Framework, can be contacted at tarma@pi.net.*

## BoundsChecker 4.0 Professional

NuMega's BoundsChecker is probably the best-known tool in its class. The BoundsChecker family started several years ago with tools for DOS and 16-bit Windows; the most recent versions (the ones examined here) target Win32 platforms. In particular, I tested BoundsChecker 4.0 Professional for Windows NT and its companion for Windows 95.

BoundsChecker can be used as a standalone program loader and tester. If you don't do anything else, BoundsChecker will intercept calls to Windows API functions and heap-related C and C++ run-time library functions while your program is running and record invalid parameters, invalid pointers, error-return codes from API functions, and a general event trace. At program termination, memory and other resource leaks are reported. If a problem is detected, BoundsChecker by default pops up a dialog box that shows the type of error, its location (if source debug information is available), and several options, among which are the abilities to suppress further reporting of the same error and to break into a debugger at the error location. All error reports are collected in BoundsChecker's log window, which can be saved at the end of the session. As a final option, BoundsChecker can perform a Win32 compliance check, which signals the presence or use (your choice) of Win32 API functions that differ among Win32s, Windows 95, and Windows NT.

If Microsoft Visual C++ 4.0 (and later) is your C++ compiler, you can take advantage of BoundsChecker's Integrated Debugging mode. In this mode, you don't need the BoundsChecker program loader but can instead use the Visual C++ workbench's debugger to run your program. The rest is the same: BoundsChecker sits in the background and pops up if it detects a problem. The resulting error log ends up in the Visual C++ Output window, under a separate BoundsChecker tab.

All this is included in the Standard edition of BoundsChecker. If you have the Professional edition, you can improve error detection by adding compile-time instrumentation (CTI). In essence, this is a preprocessing step on your C and C++ source code that adds numerous checks to pointer operations and the like that help detect several additional types of errors. Once the extra instrumentation is in place, operation is identical to the Standard edition. One thing to be aware of: CTI is currently supported only in conjunction with Microsoft Visual C++ (4.0 and later).

BoundsChecker's operation can be tweaked extensively through options accessible from its program loader (or through the Visual C++ workbench) and by means of configuration, suppression, and library specification files. Furthermore, version 4.0 adds the ability to specify custom validation modules, which let you add virtually any kind of checking or logging, using the same routines that BoundsChecker uses internally for the built-in checks.

## CodeGuard 32/16

Borland's CodeGuard is a companion tool to Borland's family of C++ compilers. It made its debut as a 16-bit add-on tool for Borland C++ 4.5 and is now part of the Borland C++ 5.0 Development Suite, with 16- and 32-bit versions covering both Win16 and Win32 programs (DOS programs are not supported by CodeGuard). During testing, I concentrated on the 32-bit version. The 16-bit version is similar, with the exception of a number of pointer checks that rely on the 32-bit CPU model and are therefore not available in 16-bit mode.

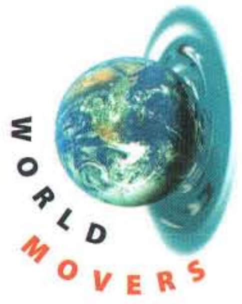
**Stake your Claim in the New World at**

# World Movers

**The VRML 2.0 Developers' Conference**

**January 30-31, 1997 at the ANA Hotel, San Francisco**

**Register Now at www.worldmovers.org**



**Secure your place at World Movers**—the VRML 2.0 event of the year. If you're a professional Web content creator or Web application developer—you won't want to miss this event. Come join the VRML revolution and help build the 3D Web.

At the conference you can join in expert panel discussions on VRML 2.0 issues such as multi-user worlds, using Java with VRML, deploying 3D games on the Net, optimizing VRML and adding life to your animations. You'll discover the right tools for efficient modeling and how to integrate VRML, HTML, Java and other digital media. And, you can attend The VRML Boot Camp—a chance to get in on the ground floor of a new industry.

In addition to the wealth of innovative sessions and the insight from industry luminaries, you'll discover exciting tools and technology from some of today's most influential companies developing for the Web.

So register for World Movers—The VRML 2.0 Developers' Conference at **www.worldmovers.org** or call **1-800-488-2883.** If you don't stake your claim now, the new world of VRML 2.0 will leave you behind.

**Our advisory board, some of the visionary leaders in the VRML industry, includes**

- **Rikk Carey,** co-author and co-architect of VRML 2.0 and an original member of the VRML Architecture Group,
- **Benjamin Feinman,** technical evangelist, Netscape Communications Corporation,
- **David Frerichs,** chairman, VRML product manager, Silicon Graphics,
- **Konstantin Guericke,** vice president of sales and marketing for Black Sun Interactive,
- **Mitra,** chief technology officer at ParaGraph International and a co-architect of VRML 2.0,
- **Tony Parisi,** chief technology officer, Intervista Software, and co-author of VRML 1.0,
- **Mark Pesce,** co-author of VRML 1.0 and an original member of the VRML Architecture Group and author of *VRML: Browsing and Building Cyberspace,*
- **James Waldrop,** technical director, Construct Inc.

**Hosted by:** SEYBOLD SEMINARS · SiliconGraphics · NETSCAPE

**Co-sponsor:** intervista

**Publication Co-sponsors:** net · PCWEEK · MacADDICT · Macworld · SILICON GRAPHICS WORLD

CIRCLE NO. 336 ON READER SERVICE CARD

The trademarks shown above are claimed, and may be registered, by their respective owners.

*(continued from page 76)*

Once installed, CodeGuard becomes part of the IDE. In fact, the C++ compiler knows enough about CodeGuard to insert the CodeGuard instrumentation code into the object code of C and C++ programs during compilation if the right options are given. With instrumentation code in place, the program is then linked to the CodeGuard library, which intercepts both C runtime and Windows API functions. At runtime, the CodeGuard DLL tracks pointer usage with the aid of the instrumented code, and API usage through the intercepted entry points. If an error is detected, CodeGuard by default pops up a message box (no specific information—just that an error was found) and writes a report to a log file. If the program is run from within the Borland IDE or Turbo Debugger, a breakpoint will occur, and the debugger will become active. At program termination, a leak search is performed and added to the log file. If so configured, CodeGuard will also add a function-call profile to the log file containing the number of times each intercepted function is called. The whole process from compilation onward can also be performed from the command line, in which case CodeGuard does report and log errors, but no breakpoints will occur.

CodeGuard's operation is configurable through a separate setup program (accessible through the IDE) or by directly editing a configuration file that records the options that apply to a given executable. Through this configuration file, you can determine the amount of memory and pointer validation, the details of API checks, and set several other options.

## Purify 4.0 NT

Pure Atria's Purify has been known for several years as a UNIX tool. With Purify 4.0 NT it is available for PC platforms, but initially only if they run Windows NT.

Purify operates as a program loader and tester. For instrumentation, it uses a technique called "Object Code Insertion" (OCI), which takes an executable module (either .EXE or .DLL), analyzes the code within, and inserts additional instructions that check pointers and memory accesses. This happens for each module that is used by a given program, including third-party modules such as system DLLs. (Don't worry, the original module is never modified; Purify works with instrumented copies.) As a result, every piece of code that is executed by your program will be instrumented, regardless of its origin. However, OCI alone does not catch all errors; to detect memory leaks, Purify uses an approach that resembles the "mark" phase in "mark and sweep" garbage-collection schemes. It recursively follows potential pointers to identify reachable heap blocks. Any blocks that cannot be reached at all are then considered leaks; any blocks that have pointers into them, but not to their start addresses, are reported as potential leaks.

After the instrumentation phase, Purify runs the instrumented program and tracks all detected errors. The error reports are collected in a log view, which is updated while the program is running. Unlike the other dynamic testers, Purify does not pop up a message box when it detects an error, but it can be configured to cause a debugger breakpoint in those cases.

Purify configuration is done from within the loader. Apart from settings that determine features such as the size of the deferred free queue and whether memory and handle leak checks are performed at program termination, Purify's output can be tailored to your needs through the use of filter sets. As its name implies, a filter determines which types of error reports are shown and which aren't. This is purely a display matter: The unwanted error reports are hidden, but remain present in the overall log. A sophisticated filter manager makes it easy to create and combine different filters and share them across programs. Finally, the Purify runtime support can be accessed through a documented API, which allows your program to communicate with Purify while it is being tested—for example, to test for new memory leaks created between two locations in your program. Incidentally, this is the only case for which you need to recompile and link your source code in order to work with Purify.

## PC-lint 7.0

Gimpel Software's PC-lint is the only test tool examined here that performs a static analysis of your program to detect potential problems. It is inspired by the well-known UNIX lint utility, but has been greatly improved over the years by Gimpel Software. Version 7.0 can produce well over 500 different diagnostics relating to C and C++ syntax, usage, and programming style, and includes fairly sophisticated techniques such as strong typing (yes, really strong, not the C/C++ idea of strong) and interstatement value tracking.

To operate PC-lint, you invoke it on your source file(s) as if it were a compiler. PC-lint parses the source code, processes include files, and so on, and complains (to *stdout*) about what it considers to be illegal, dangerous, or just bad style. Generally, it finds a lot to complain about. Its inspiration is drawn from the C and C++ (draft) standards, but also from expert advice from the books of Cargill, Coplien, Meyers, Murray, Plum, and Saks. As a result, PC-lint acts very much as a C and C++ programming-in-the-small style oracle. There is overlap with the dynamic testers, though: By virtue of its initialization and value tracking, and also because it recognizes more general problems (for example, absence of copy constructors or assignment operators in classes with point-

## Instrumentation Techniques

Instrumentation, in the context of this article, is the process of adding extra code to monitor a program's behavior. Sometimes object code in the executable image itself is changed; at other times, program flow is diverted by patching entry points to external functions.

Source-code instrumentation adds extra instructions at the source-code level. NuMega's BoundsChecker (with technology licensed from ParaSoft) uses this approach and calls it CTI ("compile time instrumentation," a slight misnomer in my view).

Compile-time instrumentation modifies the actual translation process and adds extra object code that never had a source code representation. This is what Borland's C++ compiler does for the benefit of CodeGuard.

Link-time instrumentation uses the properties of the (static) link process to intercept calls to selected library functions and replace them with calls to equivalent, but instrumented versions of them. CodeGuard uses this technique.

Object-code instrumentation takes a ready-to-run executable module and inserts additional code into it, based on an object-code-level analysis of the program flow. Pure Software's Purify is the prime example of this instrumentation technique.

Run-time instrumentation, finally, defers instrumentation to the time when the executable program image is loaded into memory, and only then modifies entry points or uses notifications (including processor exceptions) to get control at critical points. BoundsChecker uses this mode of instrumentation.

—R.v.d.W.

er data members, or absence of delete operations in destructors of the same), it will frequently spot potential memory leaks or array out-of-bound accesses without actually executing the program.

PC-lint is configurable to the extreme. You can specify options on the command line, in response files, or even embed them as comments in your source code. The options range from enabling and disabling of certain diagnostics (in general, per module or per identifier) through specification of its operating environment (to allow PC-lint to mimic, say, a Microsoft C++ compiler, complete with the right definition of _MSC_VER, integer and pointer sizes, and *include* directories) to tailoring its output format. The latter is particularly useful because it allows you to add PC-lint as a tool to your favorite IDE or editor, and then use its diagnostic output processing to jump to the right source-code location in response to PC-lint's messages. To get you started, PC-lint comes with configuration files for a few dozen C and C++ compilers.

## Evaluating the Tools

The tests I ran included several buggy programs provided by the respective vendors (obviously designed to bring out the best in their tools, and the worst in their competitors' tools), some in-depth test programs of my own, and a number of real programs that I worked on during the test period. Table 1 presents a summary of the results.

Tools are only effective if you actually use them. So, no matter how sophisticated their tests are, they must be easy to operate or they become shelfware. In fact, they should become part of the development cycle, since we all know that "testing quality into a product" as a final step in the development process is a sure way to doom your program to fail.

I could come up with carefully wrought analyses of which user interface is the best, which options make most sense, and so forth, and declare a winner on these grounds. I will not do so. Instead, I just kept a tally of how often I actually used each tool, and to which tools I turned if I had a problem. This is not quite as scientific, but it is probably more honest and more indicative of which tools stood the test of practice. You'll have to bear with my personal preferences (or aberrations) as far as my development environment goes: Throughout the test period, I mostly used Borland C++ 5.0 and Microsoft Visual C++ 4.1, and occasionally used Symantec C++ 7.21 and Watcom C++ 10.6. Programs under test were Win32 console, OWL, and MFC GUI applications, ranging from small (a few hundred lines) to

NEW

HIGH SPEED RISC I/O

**Intellicon-NT960**

***a RISC based multi-port I/O subsystem for serial communications***

**Performance**
- ★ An on-board Intel i960 32 bit RISC processor off-loads the serial I/O task from the main CPU
- ★ Offers up to 2MB of on-board dynamic RAM for data storage
- ★ Offers up to 256KB of on-board dual-ported RAM and 512KB Flash EEPROM for data storage and custom application programs
- ★ RISC-like quad UARTS with 24 bytes of FIFO per channel provide high speed data communications up to 115K baud on each port

**Hardware Flexibility**
- ★ The Intellicon-NT960 subsystem includes a NT960 host adapter, the ACM / 16 external communications module, software drivers and manuals
- ★ Provides 16 to 128 asynchronous serial ports from one PC slot
- ★ Accommodates a total of 512 serial ports with 4 host adapters in a system
- ★ The ACM / 16 module has both RJ45 and DB25 connectors
- ★ Offers optional on-board realtime clock and optional on-board battery backup for the dual-ported RAM
- ★ Supports 80286, 80386 and 80486 AT bus architecture

**Software Flexibility**
- ★ Development tools available for custom development
- ★ Software support for UNIX, XENIX, QNX, DOS

**Reliability**

Since 1985 Connect Tech's products have become known for their unsurpassed quality and reliability. This tradition of providing flexible cost effective solutions is based on Connect Tech's expertise in the design and manufacture of communication hardware and software. At Connect Tech we meet your objectives – by design.

***Call us today at telephone 519/836/1291 or facsimile 519/836/4878 to find out more about the Intellicon-NT960 and other Connect Tech communication products***

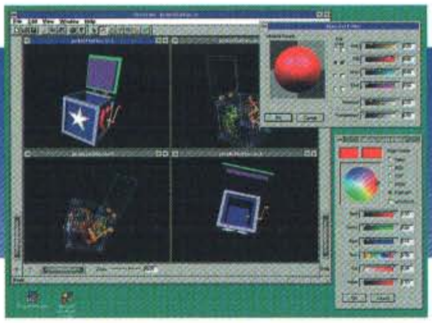
**Connect Tech Inc.**
*'Meeting your objectives ..by design'*
727 Speedvale Ave. W.
Guelph, ON N1K 1E6
Tel: 519-836-1291
Fax: 519-836-4878

CIRCLE NO. 131 ON READER SERVICE CARD

**Open Inventor**

**3D Developer's Toolkit**

Open Inventor 2.1.1 is now available for:
- Windows NT™/95
- AIX®
- Digital UNIX®
- HP-UX®
- Solaris®



- Software Updates
- Technical Support by Phone, Fax and Email

FREE for 12 MONTHS After Purchase

**The Professional 3D Graphics Programming Tool for OpenGL Development on Windows and UNIX**

Open Inventor™ redefines 3D graphics programming, making it easier to quickly prototype and develop complex, interactive 3D applications based on OpenGL®. Whether you're a novice or an expert 3D programmer, you'll find Open Inventor's extensible, object-oriented architecture to be an elegant, powerful tool for developing standards-based visualization, animation and VRML applications on Windows® and UNIX®.

**WINDOWS DEVELOPERS:** Portable Graphics' implementation of Open Inventor contains **28 new MFC extension classes,** including **OLE** support for a complete Windows *look and feel!*

**FREE Evaluation! Call Today.**

PORTABLE GRAPHICS
AN EVANS & SUTHERLAND COMPANY
3006 Longhorn, Suite 105
Austin, TX 78758
512.719.8000 • Fax: 512.832.0752
info@portable.com
http://www.portable.com

OpenGL is a registered trademark, and Open Inventor is a trademark of Silicon Graphics Inc. All other trademarks belong to their respective companies.

CIRCLE NO. 289 ON READER SERVICE CARD

| Feature | Bounds-Checker | | Code-Guard 32 | Purify NT | PC-lint |
|---|---|---|---|---|---|
| Instrumentation technique (*) | RTI | SCI | CTI | OCI | |
| Static analysis | | 0 | | | ++ |
| Read-pointer validation | | ++ | 0 | + | 0 |
| Write-pointer validation | 0 | ++ | + | ++ | 0 |
| Other pointer validation | 0 | + | + | ++ | 0 |
| Heap-based overwrites | 0 | ++ | + | ++ | |
| Stack-based overwrites | | ++ | + | | 0 |
| Global overwrites | | ++ | + | | |
| C++ checks | | ++ | ++ | + | + (static) |
| C runtime library validation | ++ | ++ | ++ | | 0 |
| Windows API validation | ++ | ++ | + | 0 | |
| OLE validation | ++ | ++ | | 0 | |
| Other API validation | + | + | | | |
| Memory-leak checks | + | ++ | + | | ++ |
| Handle in use checks | ++ | ++ | ++ | + | |
| Other resource-leak checks | + | + | + | | |
| Instrumentation speed | ++ | 0 | ++ | + | + |
| Runtime speed | + | 0 | + | +/+ | + |
| Other features | Event logging | | Call profiling | | |
| Integration with IDE | ++ (MSVC) | | ++ (BC++) | 0 (MSVC) | 0/+ |
| Borland C++ support | ≥ 4.5 | | ≥ 4.5 (**) | | Yes |
| Microsoft C++ support | ≥ 2.1 | | | ≥ 2.2 | Yes |
| Symantec C++ support | ≥ 7.0 | | | | Yes |
| Watcom C++ support | ≥ 10.5 | | | | Yes |
| Windows 95 support | Win95 version | | Yes | | Yes |
| Windows NT support | NT version | | Yes | Yes | Yes |
| Other platforms/compilers | Delphi 2.0<br>Win16 version | | Win16 | | Yes |

***Table 1:*** *Summary of features and test results. An empty cell indicates that a feature is not present; otherwise grades are 0 (some), + (good), and ++ (excellent). (*)SCI = source code, CTI = compile time, LTI = link time, OCI = object code, RTI = run time: (**) CodeGuard 32 only for Borland C++ 5.0 and later.*

large (up to 60,000 lines). All programs were written in C++.

The result: BoundsChecker (with RTI, not CTI) and Purify were the tools I used most often, with about equal frequency. The reason is quite simple: Neither requires changes to the build process (Purify's OCI is performed automatically when a program is loaded). Their usage frequencies are about the same because I use one as a second opinion to the other. I feel more comfortable with BoundsChecker's memory-leak detection and API validation, but Purify inspires more confidence in its in-depth pointer checks. On the other hand, Purify can only be used in conjunction with Microsoft-generated code, which made it unsuitable for the other compilers (which BoundsChecker could handle). The same goes for the target platform, but since I primarily use Windows NT, this was less of a problem.

BoundsChecker with CTI came in lower in the usage count for two reasons: One is that the instrumentation process requires a separate build, and the other is that the resulting executable often ran too slow for my admittedly limited patience. As a result, I used BoundsChecker Pro with CTI primarily if I needed a very thorough test; it has no equal in this respect. (By the way, this is also what NuMega recommends.) CodeGuard is also a special case. First of all, it depends on the Borland C++ compiler, and then it requires a separate build (possibly also of the libraries used by the program at hand). In the end, I mostly used it for one specific 16-bit OWL application; unfortunately, it sometimes crashed along with the program.

What about PC-lint? This is one tool that requires iron discipline to use. Despite my best intentions, I must confess that I used it far less than I planned to. Apart from my obvious lack of discipline, I attribute this to the fact that I tired of tweaking PC-lint's options over and over again. PC-lint would benefit greatly if Gimpel Software (or some kind soul in the programming community) would make an interactive option tweaker available. More than anything else, configuring PC-lint to report the important things without flooding you with minor quibbles is a chore that hampers day-to-day use of the tool. Especially if you routinely use different C++ compilers with different libraries and frameworks (as I do), you get bogged down in configuration file upon configuration file. This is a pity, because PC-lint truly deserves to be a fully integrated part of your development environment.

## Conclusions

BoundsChecker will be the testing tool of choice for many situations. It is broad in

## Static and Dynamic Testing

Static tests are performed without actually executing the program being tested; dynamic tests require execution. Program compilation is a static test; more advanced forms use proven techniques to verify the correctness of a program. Dynamic testing is performed in a crude form by protected-mode operating systems such as UNIX, OS/2, and Windows NT, which terminate an application if it steps outside its allotted address space or instruction repertoire. More advanced forms use various ways of instrumentation to keep a closer watch on the program's behavior.

The pros of static checking are that: full coverage is attainable in theory (but not yet realized in practice); it detects both faults and other problems, such as portability, style, and the like; it is independent of the quality of test cases; and error reports are immediately linked to the actual fault.

The cons of static checking are that: It requires access to source code; in practice, limits to value tracking restrict coverage; and detection of dynamic problems (for example, interactions, synchronization) is limited or absent.

The pros of dynamic checking are that: It detects problems that occur only at run time (for instance, those caused by specific interaction patterns); value tracking and API checking in principle are unlimited; and access to source code is not (always) required.

The cons of dynamic checking are that: coverage is strongly determined by the quality of actual test cases; detected failures may be difficult to relate to actual faults; instrumentation changes the program image and may introduce problems in and of itself; and run-time performance may be reduced and thereby cause problems (in real-time systems, for instance).

Regardless of the testing approach, a number of problems will not be caught. Errors of omission are notoriously hard to detect, as are errors of logic (branching the wrong way), misinterpretation of data values, and erroneous state transitions. Also, verification of a program's function against the requirements, user interface design, and performance testing are well outside the realm of these tools. Therefore, you'd be well-advised not to rely solely on automated testing tools. No matter how useful they are, a clean bill of health from such a tool should be regarded as a necessary, but by no means sufficient, condition to guarantee a correct program.

—R.v.d.W.

scope and, in combination with CTI, catches almost any error that relates to memory usage. However, you should be aware of the fact that to use CTI you need to add a separate build variant to your development process, that a CTI-instrumented executable can be significantly slower than a regular one, and finally, that only Microsoft Visual C++ compilers are supported with CTI.

CodeGuard is a good tool for Borland C/C++ users. As part of the Development Suite, it is inexpensive and covers a large number of common errors. On the other hand, it requires a separate build variant, and its error detection is sometimes flaky and may even crash the program.

Purify combines excellent error detection capabilities with an easy-to-use instrumentation step, requiring no changes to your development process. Moreover, run-time performance with instrumentation is reasonable to good, so there is little reason not to use it. Regrettably, it lacks extensive API validation and is currently only available for Microsoft Visual C++ programs running under Windows NT.

PC-lint is in a different category altogether. It can be extremely useful for both C and C++ developers, but effective use requires quite some configuration work. Nevertheless, I would strongly recommend it for use with C and C++ programming, since it will uncover much dubious or outright incorrect code that many other tools will miss.

**For More Information**

NuMega Technologies
9 Townsend West
Nashua, NH 03063
603-889-2386
http://www.numega.com/

Borland International
100 Borland Way
Scotts Valley, CA 95066
408-431-1000
http://www.borland.com/

Pure Atria
1309 South Mary Avenue
Sunnyvale, CA 94087
408-720-1600
http://www.pureatria.com/

Gimpel Software
3207 Hogarth Lane
Collegeville, PA 19426
610-584-4261
http://www.gimpel.com/

**DDJ**

# MAT<LIB>

## C++ Matrix Library

MAT<LIB> is a Matlab® Compatible C++ Matrix Class Library, designed for development of advanced scientific high-level C++ code. Evaluation version available from our home page:

http://www.mathtools.com

### *The Library*

The library includes Complex math, Binary and unary operators, Powerful indexing capabilites, Trignometric, Signal processing, File I/O, Linear algebra, String operations and Graphics. Over 300 mathematical functions are included in MAT<LIB>. Complex matrices are fully supported. Matrix can be complex or real; real matrix uses half the memory.

### *Memory*

MAT<LIB> supports matrices of doubles, floats, ints and chars mixed in the program. Images can be stored in matrices of chars, using 1/8 memory storage. On many applications, where 8 digits of precision are sufficient, float-precision matrices can save half the memory usage. Memory allocation and de-allocation is managed automatically.

### *Performance*

The library is built locally by your optimizing C++ compiler, for best performance. Local BLAS 1 and 3 libraries will be utilized.

### *Portability*

MAT<LIB> supports Windows 3.11, Windows 95, Windows NT, OS/2, Linux, SunOS, Solaris, Irix, AIX, HPUX, OSF1, Ultrix and more.

### *Linear algebra*

The Linear Algebra functions are based on the well known LINPACK and EISPACK. Cholesky, Hessenberg, LU, QR, QZ, Schur and SVD matrix decompositions are supplied.

### *Evaluation version*

Fully functional, time limited (30 days) evaluation version of MAT<LIB> can be downloaded freely from our home page.

### *Graphics*

2D and 3D Graphics are supported in standalone applications using the freely available graphics package, Gnuplot.

### *Pricing*

The price of MAT<LIB> for a single user license is $399 for commercial, $199 for academic institutes. Prices are valid until July 1, 1997.

**MathTools**

Web: http://www.mathtools.com
Email: info@mathtools.com
Fax: 1-888-MATHTOOLS

Matlab is a registered trademark of The MathWorks, Inc. Trademarks of companies mentioned appear for identification purposes only and are the property of their respective companies.

CIRCLE NO. 341 ON READER SERVICE CARD

## Listing One

```
#######################################################################
# Makefile with automated regression testing
# Copyright 1996 Adrian C. McCarthy
#######################################################################
# Written for Borland MAKE. This copy has been simplified to demonstrate
# the integration of unit testing and automated regression testing. It
# is not a complete platform for building a program.
#######################################################################
# Targets
#  accept   copies test results to reference directory
#  all      builds all executables (default)
#  clean    deletes all non-source files (executables, objs, etc.)
#  regress  builds all unit test and creates a regression report
#  release  PKZIPs the deliverable files and puts them in BINDIR
# Option Macros
#  DEBUG  if defined, compile and link with debugging information
#  MODEL  memory model to compile for
#         (l, m, c, or s for large, medium, compact, or small)
#  PLATFORM 2, 3, 4, or 5 for 286, 386, 486, or Pentium
# Directory Macros
#  BINDIR directory to place executables
#  TINDIR directory containing module test data
#  OBJDIR directory to place object files and libraries
#  OUTDIR directory to place test results
#  REFDIR directory where reference test results are kept
#  SRCDIR directory containing sources
#######################################################################

# Target platform
!ifdef PLATFORM
  PLATFORM=-$(PLATFORM)
!endif

# Debug control
!ifdef DEBUG
  CDBGOPT = -v -DDEBUG -w
    # -v       include debug info
    # -DDEBUG  #defines DEBUG
    # -w       display all warnings
  LDBGOPT = /v
  !message Debugging enabled -- no optimization
!else
  CDBGOPT = -Ox $(PLATFORM)
  !message Optimization enabled -- no debugging
!endif
!ifndef BINDIR
  BINDIR = ..\bin
!endif
!ifndef TINDIR
  TINDIR = ..\test\in
!endif
!ifndef OBJDIR
  OBJDIR = ..\obj
!endif
!ifndef OUTDIR
  OUTDIR = ..\test\out
!endif
!ifndef REFDIR
  REFDIR = ..\test\ref
!endif
!ifndef SRCDIR
  SRCDIR = ..\source
!endif

# Memory model
!ifndef MODEL
  MODEL = s
  !message Using small memory model by default.
!endif

####### Tools and Options #######
CMDLOPT = -m$(MODEL) -I$(INCLUDE)
STARTUP = c0$(MODEL).obj
RUNTIME = emu.lib math$(MODEL).lib c$(MODEL).lib
# Borland C/C++ compiler
CC = bcc -c
COPTS = $(CDBGOPT) $(CMDLOPT)
# Borland TLINK
LINK = tlink
LOPTS = /c /m /n /Tde /L$(LIB);$(OBJDIR) $(LDBGOPT)
# File Compare -- Use your favorite DIFF program here!
DIFF = fc

####### Paths #######
.path.exe=$(BINDIR)
.path.map=$(OBJDIR)
.path.lib=$(OBJDIR)
.path.obj=$(OBJDIR)
.path.c  =$(SRCDIR)
.path.cpp=$(SRCDIR)
.path.in =$(TINDIR)
.path.out=$(OUTDIR)
.path.rpt=$(OUTDIR)

####### Files #######
# Note, for each new unit test, you need to add a $(DIFF) line to the
# regress.rpt target.  You also need to add an explicit link rule for
# the corresponding unit test.  This is because the $** and $? macros
# don't work in implicit rules nor do they work with macro modifiers.
TESTEXES = tvector4.exe tmatrix4.exe tprepro.exe
TESTOUTS = tvector4.out tmatrix4.out tprepro.out
REGRPT = $(OUTDIR)\regress.rpt
.precious $(TESTOUTS) $(REGRPT)

####### Rules #######
.c.obj:
    $(CC) $(COPTS) -n$(OBJDIR) $<
.cpp.obj:
    $(CC) $(COPTS) -n$(OBJDIR) $<
# This rule runs a unit test.
.in.out:
    $(BINDIR)\$&.exe < $< > $@

####### Pseudo-targets #######
all : $(EXES) $(REGRPT)

# The accept target copies the current test results to the reference
# directory.  To avoid costly accidents, we generally keep the
# reference copies read-only, and we make backups of the current ones
# right before replacing them.
accept : $(TESTOUTS)
  -attrib -r $(REFDIR)\*.bak
  -del $(REFDIR)\*.bak
  rename $(REFDIR)\*.out *.bak
  copy $(OUTDIR)\*.out $(REFDIR)
  attrib +r $(REFDIR)\*.out
clean :
  -del $(OBJDIR)\*.obj
  -del $(BINDIR)\*.exe
  -del $(BINDIR)\*.map
  -del $(OUTDIR)\*.out
  -del $(REGRPT)
regress : $(REGRPT)
release : $(EXES)
    $(ZIP) $(BINDIR)\ART96.ZIP $(EXES)

####### Regression testing #######
#  The macro modifiers don't work on the $** or $? targets, so
#  you have to add a $(DIFF) line for each new unit test.
$(REGRPT) : $(TESTOUTS)
  echo Regression Test Report > $(REGRPT)
  echo ====================== >> $(REGRPT)
  $(DIFF) $(REFDIR)\tvector4.out $(OUTDIR)\tvector4.out >> $(REGRPT)
  $(DIFF) $(REFDIR)\tmatrix4.out $(OUTDIR)\tmatrix4.out >> $(REGRPT)
  $(DIFF) $(REFDIR)\tprepro.out  $(OUTDIR)\tprepro.out  >> $(REGRPT)

####### Dependencies #######
vector4.obj : vector4.cpp vector4.h
tvector4.obj : tvector4.cpp xbase.h vector4.h
tvector4.exe : tvector4.obj vector4.obj
  $(LINK) $(LOPTS) @&&¦
$(STARTUP)+
$**
$*.exe
$*.map
$(RUNTIME)
¦
tvector4.out : tvector4.exe tvector4.in
matrix44.obj : matrix44.cpp matrix44.h
tmatrix4.obj : tmatrix4.cpp xbase.h matrix44.h
tmatrix4.exe : tmatrix4.obj matrix44.obj vector4.obj
  $(LINK) $(LOPTS) @&&¦
$(STARTUP)+
$**
$*.exe
$*.map
$(RUNTIME)
¦
tmatrix4.out : tmatrix4.exe tmatrix4.in
prepro.obj : prepro.cpp prepro.h
tprepro.obj : tprepro.cpp prepro.h
tprepro.exe : tprepro.obj prepro.obj
  $(LINK) $(LOPTS) @&&¦
$(STARTUP)+
$**
$*.exe
$*.map
$(RUNTIME)
¦
tprepro.out : tprepro.exe tprepro.in
```

## Listing Two

```
// Unit test for Vector4.cpp. Reads a file of n vectors, exercises the Vector4
// functions on those vectors, and outputs the results.  The input file should
// start with an integer (the number of vectors) followed by the vectors.

#include <iostream.h>
#include "vector4.h"

#define TF(x)  (x ? "true " : "false")

main()
{
    int count, i, j;
    cin >> count;  // number of vectors to read from cin
    Vector4 *pV = new Vector4[count];
    Vector4 temp;

    cout << "VECTOR4 Regression Test" << endl
         << "=======================" << endl << endl;

    cout << "Reading " << count << " vectors." << endl;

    // The Vector4 module supports iostream input and output which
    // must be tested, but is also very convenient for this unit test
    // in general.
    cout << endl
         << "I/O Test" << endl
         << "========" << endl;
    for (i = 0; i < count; i++) {
        cin >> pV[i];
        cout << "I/O  " << i << "  " << pV[i] << endl;
    }
    cout << endl
         << "Unary Functions" << endl
         << "===============" << endl;
```

(continued on page 84)

IF **WEB** PROFESSIONALS **HAD**
**REGULAR JOBS,** MAYBE

WE **WOULD** HAVE **CREATED**
A **REGULAR EVENT.**

SPONSORED BY:

WEBTechniques

Web Review

CO-SPONSORED BY:

NETObjects

Microsoft

CENTURA SOFTWARE CORPORATION

IBM

intel

Powersoft

SYBASE

## WEB DESIGN & DEVELOPMENT '97
**FEBRUARY** 22–26, 1997, **MOSCONE,** SAN FRANCISCO

You know it and we know it — building useful web sites requires some pretty remarkable skills. To do it, you need to balance technical brilliance, design genius, and marketing savvy. To aid your cause, we've created the industry's deepest platform- and product-independent conference, structured around the five stages of web design and development. We've enlisted web giants in VRML, Intranets, Java, interface design, animation, databases, cybercash and more as keynotes and instructors. We've amassed a massive tools exhibition. And we've added some elements that make Web '97, like the web itself, downright exhilarating. So if you're committed to doing something spectacular with your web site, you'll certainly want to explore ours to learn more and register. BIG. **VERY BIG.**

**WEB** DESIGN & DEVELOPMENT

www.web97.com

Feeling nostalgic? Contact us by phone at 415.905.2702 or by e-mail at web97@mfi.com

**Communications?**

**Compression?**

**Lots of New Stuff From Greenleaf**

*Visit us today...*

http://www.gleaf.com/~gleaf

**NEW**—ViewComm for Windows

Asynch communications debugger & protocol analyzer for Windows 95 and NT. See data and control signals on RS-232 links in real-time or review captured data. Flexible trigger system provides start-stop control of display or capture. Graphical display of modem & control signals. Data display in hex, octal, decimal, binary. ASCII & EBCDIC data. Character or hex displays. Searches. Many extras. ViewComm/DOS also available. Download TestDrive today: http://www.gleaf.com/~gleaf

**NEW**—ArchiveLib 2.1

Greenleaf proprietary and PKZIP 2.0x compatible compression engines. Flexible archiving including compatibility with PKZIP. Archive to memory or file. PKWARE compatible encryption. Visual Basic, Pascal, Delphi, and C/C++ APIs. Supports Windows 95, NT, Windows 3.1x, OS/2 Warp, and 16- and 32-bit DOS.

Greenleaf Comm++ 3.0

Async communications the C++ way. Classes for file transfer, terminal emulation, screen drivers, modem and port control, flow control. Supports 16550, smart and non-smart multi-port boards. Supports Windows 95, NT, Windows 3.1x, OS/2 Warp, Novell NASI and 32-bit DOS.

*No royalties. Lots of examples. Onscreen documentation. Library products come with free source code. 30-day no-questions money-back. Download free test-drive software.*

*Call today for complete information, demo, or to order. MasterCard, VISA, AmEx and Novus; approved PO's*

**1-800-523-9830**

Greenleaf Software, Inc.
16479 Dallas Parkway, Suite 570
Dallas, TX 75248
800.523.9830 972.248.2561
FAX 972.248.7830
BBS 972.250.3778 14.4/8/N/1

**http://www.gleaf.com/~gleaf**
email info@gleaf.com

GREENLEAF Software®

**CIRCLE NO. 87 ON READER SERVICE CARD**

*(continued from page 82)*

```
    for (i = 0; i < count; i++) {
        cout << pV[i] << " Homogeneous:  "
             << TF(pV[i].IsHomogeneous()) << endl;
        cout << "Homogenize " << pV[i] << " == ";
        pV[i].Homogenize();
        cout << pV[i] << endl;

        cout << pV[i] << " Unit:  " << TF(pV[i].IsUnit()) << endl;
        temp = pV[i];
        if (temp.Magnitude() != 0.0) {
            temp.Unitize();
            cout << "Unitize " << pV[i] << " == " << temp
                 << " (" << temp.Magnitude() << ")" << endl;
        }
        else {
            cout << "Can't make zero vector unit length." << endl;
        }
        cout << "||" << pV[i] << "||   = " << pV[i].Magnitude()
             << endl;
        cout << "||" << pV[i] << "||^2 = " << pV[i].MagSquare()
             << endl;
        temp = -pV[i];
        cout << " -" << pV[i] << " = " << temp << endl;
        temp = 0*pV[i];
        cout << "0*" << pV[i] << " = " << temp << endl;
        temp = 1.0*pV[i];
        cout << "1*" << pV[i] << " = " << temp << endl;
        temp = pV[i]*2;
        cout << pV[i] << "*2"    " = " << temp << endl;
        cout << "---" << endl;
    }
    cout << endl
         << "Binary functions" << endl
         << "================" << endl;
    for (i = 0; i < count; i++) {
        for (j = 0; j < count; j++) {
            cout << pV[i] << " == " << pV[j] << ":  "
                 << TF(pV[i] == pV[j]) << endl;
            cout << pV[i] << " != " << pV[j] << ":  "
                 << TF(pV[i] != pV[j]) << endl;
            temp = pV[i] + pV[j];
            cout << pV[i] << " + " << pV[j] << " = " << temp << endl;
            temp = pV[i] - pV[j];
            cout << pV[i] << " - " << pV[j] << " = " << temp << endl;
            temp = pV[i];
            temp += pV[j];
            cout << pV[i] << " += " << pV[j] << " = " << temp
                 << endl;
            temp = pV[i];
            temp -= pV[j];
            cout << pV[i] << " -= " << pV[j] << " = " << temp << endl;
            cout << pV[i] << " dot " << pV[j] << " = "
                 << Dot(pV[i], pV[j]) << endl;
            temp = Cross(pV[i], pV[j]);
            cout << pV[i] << " cross " << pV[j] << " = " << temp
                 << endl;
            cout << "---" << endl;
        }
    }
    delete pV;
    return 0;
}
```

## Listing Three

```
9
[1 0 0]
[0 1 0]
[0 0 1]
[1 1 1]
[0 0 0]
[4 4 4 (2)]
[1 2 3 (1)]
[3 2 1 (0)]
[1.234567 1.23456789 1.2345678901234567890]
```

## DEBUG/TRACE

## Listing One

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

#define INF 1000        /*Infinity must be higher than the largest path  value */
#define N     10        /*Maximum number of nodes in the network */
#define TRUE   1
#define FALSE  0
#define PERM   1        /*Permanently labeled */
#define TEMP   0        /*Temporarily labeled */

/*--------Function declarations------------------------------------*/

int dijkstra(int s, int t, int w[][N], int final[], int dist[], int pred[]);
extern void dbg_init(void);
extern void dbg(char *, int, char *, ...);

/*--------Function definitions-------------------------------------*/
int dijkstra(int s, int t, int w[][N], int final[], int dist[], int pred[])
{
  int u, v, y, recent, newlabel, min, path;

/*--------Initialization----------------------------*/
dbg_init();
  for (v=0; v<N; v++)
  {
    dist[v]  = INF;   /*Set all distances to infinity */
    final[v] = TEMP;  /*Set all nodes to "temporarily labeled" */
    pred[v]  = -1;    /*Set nodes in shortest path trace to "non existent" */
  }
  dist[s]  = Ø;       /*Distance from s to s is Ø */
  final[s] = PERM;    /*Set node s to "permanently labeled" */
  path     = TRUE;    /*Assume there is a path from s to t */
  recent   = s;       /*Most recent node is s */


/*--------Run Dijkstra's Algorithm------------------*/
  for(u=Ø;u<6;u++)
  {
     for(v=Ø;v<6;v++)
       dbg("1_init1",2,"w[/d][/d] = /d   ",u,v,w[u][v]);
  }
  dbg("1_init2",1,"&u = /p\n",&u);
  while(final[t]==TEMP)   /*while destination node t is temporarily labeled */
  {
     for(v=Ø; v<N; v++)
     {
    if ((w[recent][v] < INF) && (final[v]==TEMP)) /*node temporarily labeled */
        {
           newlabel = dist[recent] + w[recent][v];
           if (newlabel < dist[v]) /*new distance, smaller than previous one */
           {
          dist[v] = newlabel;   /*update distance */
          pred[v] = recent;     /*put predecessor in shortest path */
          dbg("2_dist",4,"recent = /d   newlabel = /d \n",recent,newlabel);
           }
        }
     }
     min = INF;             /*reset temporary minimum */
     for(u=Ø; u<N; u++)   /*Find smallest labeled node */
     {
    dbg("3_u",1,"u=/d\n",u);
        if ((final[u]==TEMP) && (dist[u] < min))  /*node temporarily labeled */
        {
           y   = u;       /*Save node in y */
           min = dist[u]; /*Reduce temporary minimum */
        }
     }
     if (min < INF)       /*if there is a path */
     {
        final[y] = PERM;
        recent   = y;
     }
     else                 /*if there is no path */
     {
        path     = FALSE;
        final[t] = PERM;  /*induce break of while loop */
     }
  }
  return(path);
}
```

## NT FILE SYSTEM

### Listing One

```
typedef struct _FILE_OBJECT {
    CSHORT           Type;
    CSHORT           Size;
    PDEVICE_OBJECT   DeviceObject;
    PVPB             Vpb;
    PVOID            FsContext;
    PVOID            FsContext2;
    PSECTION_OBJECT_POINTERS SectionObjectPointer;
    PVOID            PrivateCacheMap;
    NTSTATUS         FinalStatus;
    struct _FILE_OBJECT *RelatedFileObject;
    BOOLEAN          LockOperation;
    BOOLEAN          DeletePending;
    BOOLEAN          ReadAccess;
    BOOLEAN          WriteAccess;
    BOOLEAN          DeleteAccess;
    BOOLEAN          SharedRead;
    BOOLEAN          SharedWrite;
    BOOLEAN          SharedDelete;
    ULONG            Flags;
    UNICODE_STRING   FileName;
    LARGE_INTEGER    CurrentByteOffset;
    ULONG            Waiters;
    ULONG            Busy;
    PVOID            LastLock;
    KEVENT           Lock;
    KEVENT           Event;
    PIO_COMPLETION_CONTEXT CompletionContext;
} FILE_OBJECT;
```

### Listing Two

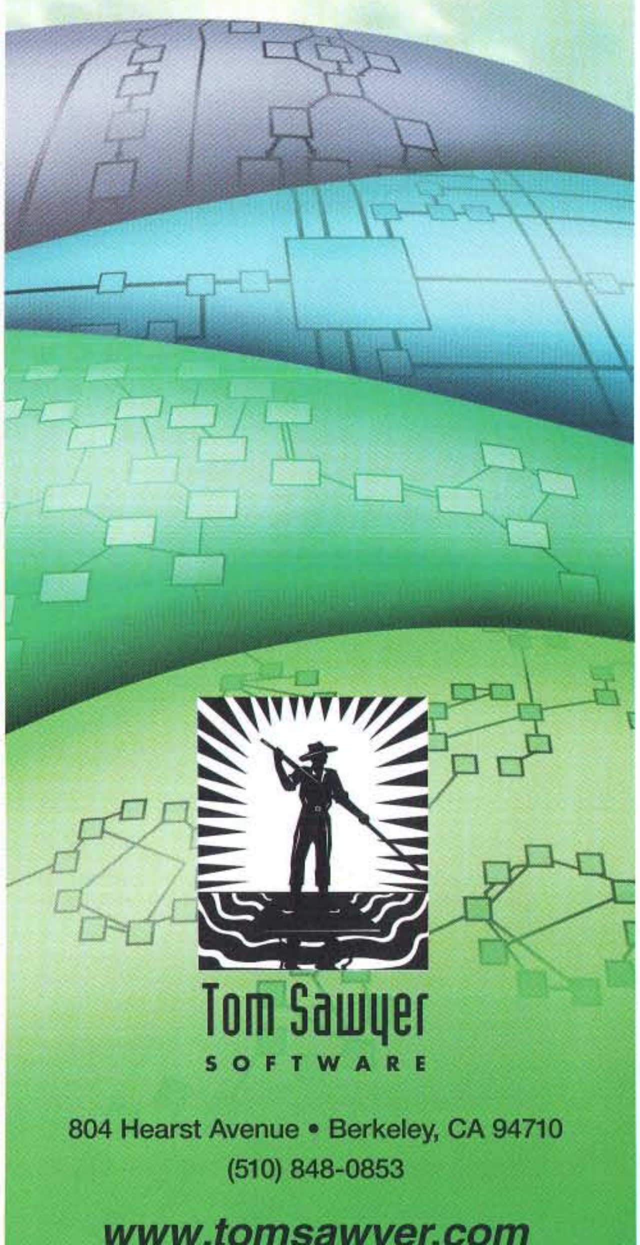
```
typedef struct _IRP {
    CSHORT           Type;
    USHORT           Size;
    // Define the common fields used to control the IRP.
    // Define a pointer to the Memory Descriptor List (MDL) for this I/O
    // request.  This field is only used if the I/O is "direct I/O".
    PMDL             MdlAddress;
    // Flags word - used to remember various flags.
    ULONG            Flags;
    // The following union is used for one of three purposes:
    //  1. An associated IRP. Field is a pointer to a master IRP.
    //  2. The master IRP. Field is the count of the number of IRPs which must
    //       complete (associated IRPs) before the master can complete.
    //  3. This operation is being buffered and the field is the address of
    //       the system space buffer.
    union {
        struct _IRP      *MasterIrp;
        LONG             IrpCount;
```

*(continued on page 86)*

# GRAPH LAYOUT TOOLKIT

***Relationship***

***Visualization***



**Tom Sawyer**
**SOFTWARE**

804 Hearst Avenue • Berkeley, CA 94710
(510) 848-0853

***www.tomsawyer.com***

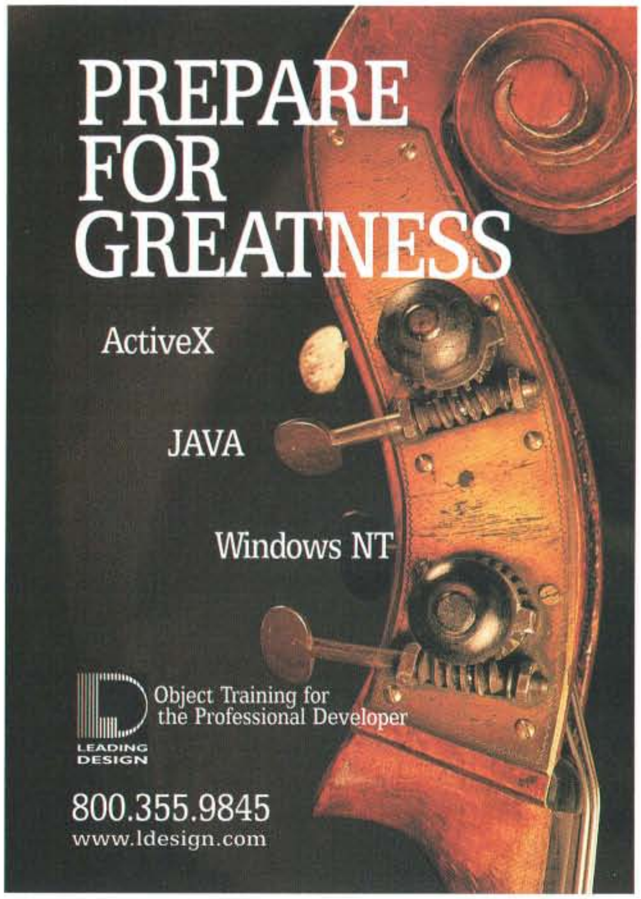
**CIRCLE NO. 301 ON READER SERVICE CARD**

*(continued from page 85)*

```
        PVOID           SystemBuffer;
    } AssociatedIrp;

    // Thread list entry - allows queueing the IRP to the thread pending I/O
// request packet list.
    LIST_ENTRY          ThreadListEntry;
    // I/O status - final status of operation.
    IO_STATUS_BLOCK     IoStatus;
// Requestor mode - mode of the original requestor of this operation.
    KPROCESSOR_MODE RequestorMode;
    // Pending returned - TRUE if pending was initially returned as the
    // status for this packet.
    BOOLEAN            PendingReturned;
    // Stack state information.
    CHAR               StackCount;
    CHAR               CurrentLocation;
    // Cancel - packet has been canceled.
    BOOLEAN            Cancel;
    // Cancel Irql - Irql at which the cancel spinlock was acquired.
    KIRQL              CancelIrql;
    // ApcEnvironment - Used to save the APC environment at the time that the
    // packet was initialized.
    CCHAR              ApcEnvironment;
    // Zoned - packet was allocated from a zone.
    BOOLEAN            Zoned;
    // User parameters.
    PIO_STATUS_BLOCK UserIosb;
    PKEVENT            UserEvent;
    union {
        struct {
            PIO_APC_ROUTINE UserApcRoutine;
            PVOID       UserApcContext;
        } AsynchronousParameters;
        LARGE_INTEGER   AllocationSize;
    } Overlay;
    // CancelRoutine - Used to contain the address of a cancel routine supplied
    // by a device driver when the IRP is in a cancelable state.
    PDRIVER_CANCEL      CancelRoutine;
    // Note that the UserBuffer parameter is outside of the stack so that I/O
    // completion can copy data back into the user's address space without
    // having to know exactly which service was being invoked.  The length
    // of the copy is stored in the second half of the I/O status block. If
    // the UserBuffer field is NULL, then no copy is performed.
    PVOID            UserBuffer;
    // Kernel structures
    // The following section contains kernel structures which the IRP needs
    // in order to place various work information in kernel controller system
    // queues.  Because the size and alignment cannot be controlled, they are
    // placed here at the end so they just hang off and do not affect the

    // alignment of other fields in the IRP.
    union {
        struct {
            // DeviceQueueEntry - The device queue entry field is used to queue
            // the IRP to the device driver device queue.
            KDEVICE_QUEUE_ENTRY DeviceQueueEntry;
            // Thread - pointer to caller's Thread Control Block.
            PETHREAD            Thread;
            // Auxillary buffer - pointer to any auxillary buffer that is
// required to pass information to a driver that is not contained
            // in a normal buffer.
            PCHAR           AuxiliaryBuffer;
            // List entry - queues packet to completion queue, among others.
            LIST_ENTRY      ListEntry;
            // Current stack location - contains a pointer to the current
            // IO_STACK_LOCATION structure in the IRP stack.  This field
            // should never be directly accessed by drivers.  They should
            // use the standard functions.
            struct _IO_STACK_LOCATION *CurrentStackLocation;
            // Original file object - pointer to the original file object
            // that was used to open the file.  This field is owned by the
            // I/O system and should not be used by any other drivers.
            PFILE_OBJECT        OriginalFileObject;
        } Overlay;
        // APC - This APC control block is used for the special kernel APC as
        // well as for the caller's APC, if one was specified in the original
        // argument list.  If so, then the APC is reused for the normal APC for
        // whatever mode the caller was in and the "special" routine that is
        // invoked before the APC gets control simply deallocates the IRP.
        KAPC        Apc;
        // CompletionKey - This is the key that is used to distinguish
        // individual I/O operations initiated on a single file handle.
        //
        ULONG       CompletionKey;
    } Tail;
} IRP, *PIRP;
```

## Listing Three

```
BOOLEAN HookDrive( IN char Drive, IN PDRIVER_OBJECT DriverObject )
{
   IO_STATUS_BLOCK       ioStatus;
   HANDLE                ntFileHandle;
   OBJECT_ATTRIBUTES     objectAttributes;
   PDEVICE_OBJECT        fileSysDevice;
   PDEVICE_OBJECT        hookDevice;
   UNICODE_STRING        fileNameUnicodeString;
   WCHAR                 filename[] = L"\\DosDevices\\A:\\";
   NTSTATUS              ntStatus;
   ULONG                 i;
   PFILE_OBJECT          fileObject;
   PHOOK_EXTENSION       hookExtension;

    if ( Drive >= 'a' && Drive <= 'z' ) Drive -= 'a';
    else                  Drive -= 'A';
    if ( (unsigned char)Drive >= 26 )
    return FALSE;

    if ( LDriveDevices[Drive] == NULL )  {
```



CIRCLE NO. 150 ON READER SERVICE CARD

REACH AND KILL MEMORY BUGS IN C/C++ CODE YOU CAN'T DEBUG:
3rd party DLLs
X Windows
the C run time library
3rd party object libraries
automatically generated code
and legacy code, to name a few

- Simply link in Great Circle and watch as it quickly finds and kills all the impossible-to-reach memory bugs
- 90-day money-back guarantee
- Call for more info and your free Great Circle white paper: "Automatic Memory Management in C/C++ with Garbage Collection"

INFO WORLD TOP SCORE

SOFTWARE Development

GreatCircle

from

Geodesic systems

Phone: 800.360.8388
Fax: 773.728.6096
E-mail: info@geodesic.com

Web Site: www.geodesic.com

CIRCLE NO. 285 ON READER SERVICE CARD

```
    // point to the current logical drive
    filename[12] = 'A'+Drive;
    // have to figure out what device to hook - first open the root directory
    RtlInitUnicodeString( &fileNameUnicodeString, filename );
    InitializeObjectAttributes( &objectAttributes, &fileNameUnicodeString,
            OBJ_CASE_INSENSITIVE, NULL, NULL );
    ntStatus = ZwCreateFile( &ntFileHandle, SYNCHRONIZE¦FILE_READ_ACCESS,
           &objectAttributes, &ioStatus, NULL, Ø,
           FILE_SHARE_READ¦FILE_SHARE_WRITE, FILE_OPEN,
           FILE_SYNCHRONOUS_IO_NONALERT¦
           FILE_OPEN_FOR_BACKUP_INTENT¦FILE_DIRECTORY_FILE, NULL, Ø );
    if( !NT_SUCCESS( ntStatus ) ) return FALSE;
    // got the file handle, so now look-up the file-object
    ntStatus = ObReferenceObjectByHandle( ntFileHandle, FILE_READ_DATA,
                        NULL, KernelMode,&fileObject, NULL );
    if( !NT_SUCCESS( ntStatus ))    return FALSE;
    // now, find out what device is associated with the object
    fileSysDevice = IoGetRelatedDeviceObject( fileObject );
    if ( ! fileSysDevice ) {
        ZwClose( ntFileHandle );
        return FALSE;
    }
    // make sure we haven't already attached to this device
    // (which can happen for directory mounted drives for networks)
    for( i = Ø; i < 26; i++ ) {
        if( LDriveDevices[i] == fileSysDevice ) {
            // yep, already watching it so add it to a group
            ObDereferenceObject( fileObject );
            ZwClose( ntFileHandle );
            LDriveMap[ Drive ]     = LDriveMap[i];
            LDriveDevices[ Drive ] = fileSysDevice;
            return TRUE;
        }
    }
    // create a device to attach to the chain
    ntStatus = IoCreateDevice( DriverObject, sizeof(HOOK_EXTENSION), NULL,
               fileSysDevice->DeviceType, Ø, FALSE, &hookDevice );
    // did we create a device successfully?
    if ( !NT_SUCCESS(ntStatus) ) return FALSE;
    // clear our init flag as per NT DDK KB article on creating
    // device objects from a dispatch routine
    hookDevice->Flagsn &= ~DO_DEVICE_INITIALIZING;
    // set-up the device extensions
    hookExtension = hookDevice->DeviceExtension;
    hookExtension->LogicalDrive = 'A'+Drive;
    hookExtension->FileSystem   = fileSysDevice;
    // now, attach ourselves to the device
    ntStatus = IoAttachDeviceByPointer( hookDevice, fileSysDevice );
    if ( !NT_SUCCESS(ntStatus) )  {
        // if we couldn't attach
        ObDereferenceObject( fileObject );
        ZwClose( ntFileHandle );
        return FALSE;
    } else {
        // assign this drive a drive group if it doesn't have one
        if( !LDriveMap[ Drive ] ) LDriveMap[ Drive ] = ++LDriveGroup;
    }
    // close the file and update our list
    ObDereferenceObject( fileObject );
    ZwClose( ntFileHandle );
    LDriveDevices[Drive] = hookDevice;
    }
    return TRUE;
}
```

## C++ PROGRAM ANALYZERS

### Listing One

```
//  20  M24  S   Avoid gratuitious use of virtual inheritance, i.e., make
//  sure there are at least two inheritance paths to each virtual base class.
class Base { int x; };
class Derived: virtual public Base {};
Derived d;
```

### Listing Two

```
class B { ... };              // base class;  assume no virtual dtor
class D: public B { ... };    // derived class
void f(B *p);                 // f is some function taking a B*
D *pd = new D;                // pd points to a D
f(pd);                        // pass pd to f, binding pd to p in f
void f(B *p)
  {
    delete p;                 // this calls only B's dtor, not D's!
  }
```

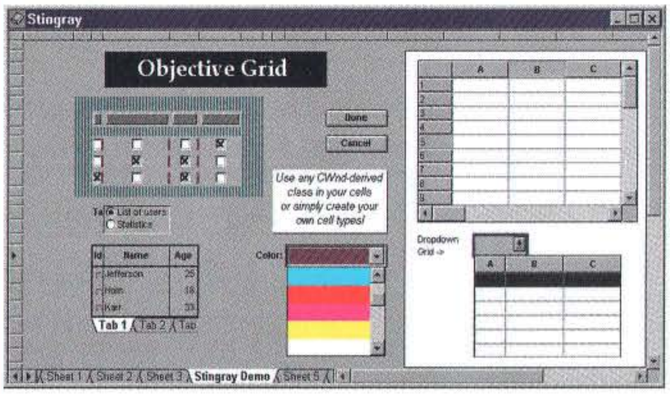
### Listing Three

```
// test program for rule 10

class Base {};
class Derived: public Base {};


int main()
{
  Base *pb = new Derived;
  delete pb;
  return 0;
}
```

**DDJ**

# Tired of Wrestling with OCX/DLLs??

*You need Stingray's 100% MFC extensions:*
***Objective Grid*™ *and* Objective Toolkit™ *1.2 !!***



**Objective Grid** (pictured) is a complete grid MFC extension. You can use the grid in a CView, CWnd and even as a popup. The grid can be bound to any external data source with one virtual override. Objective Grid also features complete ODBC/DAO support, print preview, find/replace, undo/redo, UNICODE/DBCS and an extensible control architecture.

The **Objective Toolkit** library includes over 40 MFC classes:

| | |
|---|---|
| ☆ **Docking Views** | ☆ **Tabbed Window** |
| ☆ **Calendar Control** | ☆ **Colorwell Control** |
| ☆ **Shortcut Editor** | ☆ **Encrypting CFile** |
| ☆ **Workspaces** | ☆ **Workbook MDI Alt.** |
| ☆ **Compressing CFile** | ☆ **VB-like floating windows** |
| ☆ **Mask Edit** | ☆ **Thumbnailing** |

☆ **Image classes (GIF/DIB/JPEG/PCX/TGA/TIFF)**

☆ **Plus: file system class, zooming/panning CViews, and much more!**

Both products include a 30-day money back guarantee, 60 days of technical support, no royalties and **FULL SOURCE CODE!** Visual C++ 1.5x and 4.x are supported. Objective Grid is $395 and SEC++ is $495. Annual subscriptions are available for both. All major credit cards are accepted.

**Save a bundle: Objective Grid/Toolkit bundle is only $795. Order today!**

**For FREE DEMOS Surf to:**

**WWW.STINGSOFT.COM**

Stingray™

"We add class to MFC!"™

Stingray Software, Inc.
**1-800-924-4223**
Voice: (919) 933-0863
Fax: (919) 933-0892
Email: sales@stingsoft.com

CIRCLE NO. 286 ON READER SERVICE CARD

# Dr. Dobb's CD-ROM Library

## The Software Professional's Essential Resource
## Visit our website! www.ddj.com/cdrom/

The *Dr. Dobb's CD-ROM Library* is an extensive collection of powerful tools and resources, essential to every programmer's productivity and success. The library focuses on critical aspects of programming ranging from an expansive back-issue archive of *Dr. Dobb's Journal* to essential book selections on *Algorithms* and *Data Structures* and also *Graphics Programming*. And coming from *Dr. Dobb's*, you know that each CD-ROM is chock-full of source code.

Take a moment to browse through our library. We guarantee your satisfaction.

NEW RELEASE!

### *Dr. Dobb's/CD Release 4*

- Over 8-1/2 years (Jan. 1988 to Jun. 1996) of *Dr. Dobb's Journal and Sourcebook*
- Includes every article, all the source code, and a super-fast search engine

Trusted answers are hard to come by, but not with the *Dr. Dobb's/CD!* Whether it's source code or detailed documentation on a specific subject—*Dr. Dobb's/CD* has what you need.

*Release 4* features all the articles and source code from *Dr. Dobb's Journal* from January 1988 through June 1996 (over 8-1/2 years) . . . and also includes the *Dr. Dobb's Sourcebook Series*.

Conduct full text searches with multiple keywords, Boolean logic, and wildcards . . quickly. You can copy and paste, export, and print directly from the CD-ROM. It's the ultimate programming archive!

Run your *Dr. Dobb's/CD* on either DOS or Windows.

Put the programming expertise and innovation of today's best programmers at your fingertips.

$29.95 upgrade available to prior owners!
To upgrade call: 800-822-1162

**Price: $79.95**

***FAST*** **Search Engine**

### *Hands-On PostScript Programming CD-ROM*

NEW RELEASE!

Everything you need to get started with the language of PostScript Programming is contained in this new release from the *Dr. Dobb's CD-ROM Library* entitled *Hands-On Postscript Programming*. The CD-ROM includes the full text of Gunter Born's, *Programming in PostScript*, which contains comprehensive descriptions and documentation on Postscript Programming, as well as the following tools and utilities; the Aladdin GhostScript 4.03 PostScript Interpreter, Adobe Acrobat, conversion programs, samples, and plenty of source code. This CD-ROM will allow you to become a proficient PostScript programmer.

The *Hands-On Postscript Programming CD-ROM* supports the following operating systems; DOS, Macintosh, OS/2, UNIX, Windows 3.1, Windows 95, Windows NT, and VMS.

**Price: $39.95**

## Order Today!

**U.S. & Canada:**
**800-992-0549**
**All Other Countries:**
**913-841-1631**

**E-mail:**
**orders@mfi.com**
**Fax Orders:**
**913-841-2624**

**Mail Orders**
Dr. Dobb's CD-ROM Library
1601 West 23rd St., Ste. 200
Lawrence, KS 66046-2700 USA

**Go to our website for more products from Dr. Dobb's CD-ROM Library: www.ddj.com/cdrom/**

Dr. Dobb's CD-ROM LIBRARY

## Buy 2 CDs and get 25% off the 3rd!

### *Essential Books on File Formats CD-ROM*

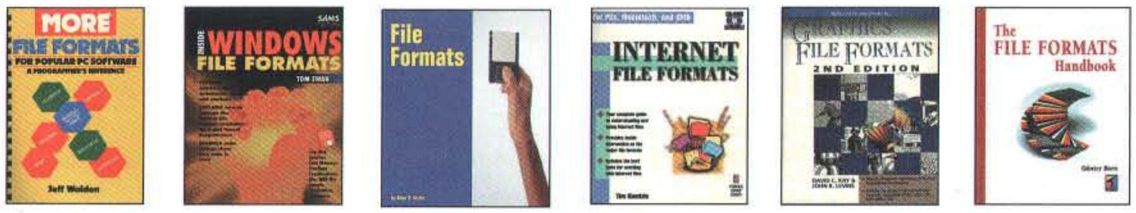
NEW RELEASE!

• 6 Essential Books selected by the *Dr. Dobb's* editorial staff

Look no further! The *Essential Books on File Formats CD-ROM* contains the complete text from six books, which will provide you with the most comprehensive and detailed information on all the important and popular file formats in use today. Selected by the editors of *Dr. Dobb's*, this CD-ROM contains invaluable information on file formats used for graphics, multimedia, sound, databases, spreadsheets, Windows, the Internet, and much more!

No matter what your programming focus, the *Essential Books on File Formats CD-ROM* discloses the secrets and insider knowledge you need to make your programming instantly compatible with all the major applications out there right from the start. Plus, the CD-ROM's powerful full-text search engine and hyperlink capabilities allow you to search quickly and easily across all the books to link directly to the information you need.

**Price: $69.95**



**FAST Search Engine**

### *Essential Books on Algorithms and Data Structures CD-ROM*

• 8 essential books selected by the *Dr. Dobb's* Editorial Staff

Have you ever been stumped when creating an algorithm or data structure? Then the *Essential Books on Algorithms and Data Structures CD-ROM* belongs on your desktop!

These eight books, selected by the *Dr. Dobb's* editors, are the most important books every written on the subject and provide you, the software professional, with the details you need to succeed. Plus, this CD-ROM features a full-text search engine, giving you instant access to the entire contents of all eight books.

Learn from the experts. Set up your algorithms the right way, the first time—before you ever write that first line of code!

Your CD-ROM includes these must-have features:

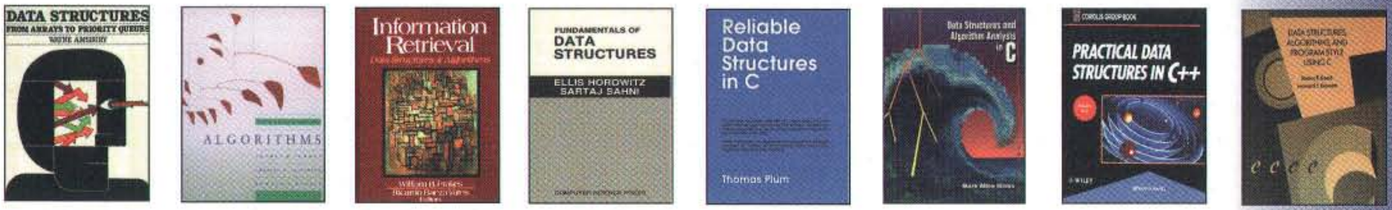
- A Powerful Full-Text Search Engine
- Full Windows Functionality
- Hyperlinked TOCs and Indexes for Each Book
- Create Your Own Bookmarks and Annotations
- Plus Much More!

**Price: $59.95**

***8 Complete Books On One CD-ROM!***

- *Data Structures: From Arrays to Priority Queues,* by Wayne Amsbury
- *Introduction to Algorithms,* by Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest
- *Information Retrieval: Data Structures & Algorithms,* edited by William B. Frakes and Ricardo Baeza-Yates
- *Fundamentals of Data Structures,* by Ellis Horowitz and Sartaj Sahni
- *Reliable Data Structures in C,* by Thomas Plum
- *Data Structures and Algorithm Analysis in C,* by Mark Allen Weiss
- *Practical Data Structures in C$^{++}$,* by Bryan Flamig
- *Data Structures, Algorithms and Program Style Using C,* by James F. Korsh and Leonard J. Garrett

**FAST Search Engine**


DATA STRUCTURES FROM ARRAYS TO PRIORITY QUEUES
ALGORITHMS
Information Retrieval
FUNDAMENTALS OF DATA STRUCTURES
Reliable Data Structures in C
PRACTICAL DATA STRUCTURES IN C++

## Order Today!

**U.S. & Canada:**
**800-992-0549**
**All Other Countries:**
**913-841-1631**

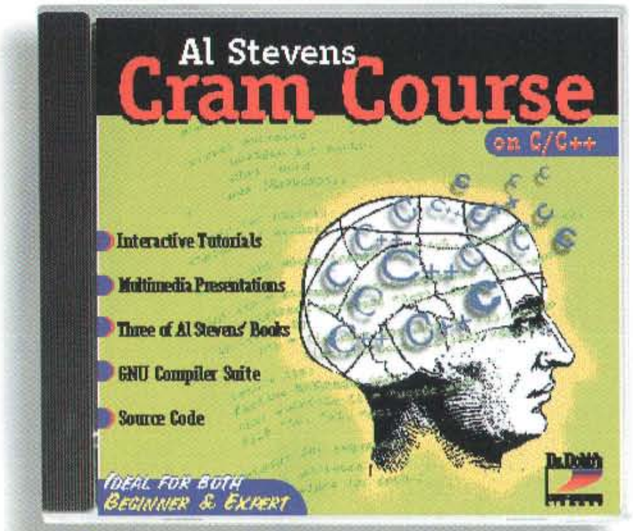
**E-mail:**
**orders@mfi.com**
**Fax Orders:**
**913-841-2624**

**Mail Orders**
Dr. Dobb's CD-ROM Library
1601 West 23rd St., Ste. 200
Lawrence, KS 66046-2700 USA

**Go to our website for more products from the Dr. Dobb's CD-ROM Library: www.ddj.com/cdrom/**

# Dr. Dobb's CD-ROM Library presents Al Stevens Cram Course on C/C++

THE C/C++ PROGRAMMERS ESSENTIAL RESOURCE



Al Stevens, a world renowned programming expert and contributing editor for *Dr. Dobb's Journal*, has created the CD-ROM to answer all your C/C++ programming questions – the *Al Stevens Cram Course on C/C++*.

This CD-ROM includes the complete text of three books written by Al Stevens, an interactive step-by-step tutorial with precise explanations, video clips of Al discussing important topics, a compiler directly connected to the exercises, lots of usable source code, plus a memory feature that bookmarks your place for quick returns to prior sessions.

Use this CD-ROM on your programming projects to answer questions, work through actual examples, compile your example source code, and make sure it's correct right then and there.

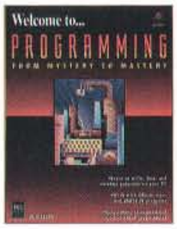
No matter what your level of programming expertise, the *Al Stevens Cram Course on C/C++* will help you become a more proficient programmer.

**This multimedia CD-ROM features:**

- Three Complete Programming Books
- Interactive Tutorials
- Built in Compiler
- Video Clips
- Bookmark for Quick Returns
- Background Music Performed by Al Stevens

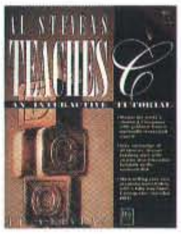
See screen shots and a more detailed explanation on our Website: www.ddj.com/cdrom/alstevens.htm

**Includes these Books by Al Stevens:**



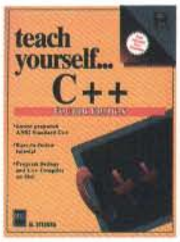
**Welcome to Programming**

*Straight forward explanations of computer files, code, and language makes programming simple.*



**Al Stevens Teaches C**

*The perfect introduction to C with an organized, structured approach, teaching good programming through good example.*



**Teach Yourself ...C++**

*This book demonstrates the hows and whys of C++ program design in an accessible, informative style.*

Dr.Dobb's CD-ROM LIBRARY

**Or complete the form and return to :**
**Fax: 913-841-2624; E-mail: orders@mfi.com;**
**Int'l: Use mail, fax, e-mail, or call 913-841-1631**

**Please send me an *Al Stevens Cram Course on C/C++* CD-ROM today!** My price is $69.95 (plus $2.00 U.S./Canada; all other countries $10.00). Please add sales tax in the following states: CA (8.5%); GA (6%); IL (6.25%), KS (6.9%), NY (8.25%), TX (8.25%)

Name

Address

City | State | ZIP | Country

Phone | ☐ Check ☐ VISA ☐ Mastercard ☐ Amex

Credit Card#: | Exp. Date

Signature

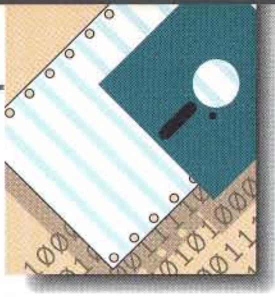
J0297

## Order Today! 800-548-1971

SYSTEM REQUIREMENTS:

- 486/40 (486/66 recommended)
- SVGA
- CD-ROM
- Mouse
- 8 MB RAM
- Win95 or NT
- Sound Card (Not essential)
- 5 MB Disk Space

(Will run on Windows 3.1, but requires manual installation)



# Some Observations on Apple and Java

Michael Swaine

Over the years I have tried, in print here and elsewhere, to follow the evolution of the Macintosh operating system. I haven't written much on the subject recently, because there just hasn't been much to write. This month, though, I face the formidable challenge of writing about the future of the MacOS after it has more or less been pronounced officially dead.

After that exercise in exhumation, I'll be in just the right mood to dig up some more bones from computing's history next month. This month, though, I'm playing devil's advocate by: 1. assuming Apple has a future, and 2. criticizing Java.

### Beaten: The System

In the only slightly distant past, just this side of where the mists of forgetfulness drift as thick as my beard and as gray as my hair, I wrote, for a magazine that shall remain nameless, a column that shall be here named "Beating the System."

The system in question was the MacOS, the magazine in question was one dedicated to the Mac user, and in those now-hazy days (though I'm not so sure they weren't just as hazy when we were going through them) the MacOS was a mere rosy-cheeked six-dot-ecks versions old. I share these reminiscences with you not only out of the magnanimity of my spirit but also in furtherance of a point, which I expect we'll stumble over in one of the forthcoming paragraphs.

Earlier, I had done a HyperTalk scripting column for the magazine in question, and later, toward the end of my tenure as a system beater for the mag in Q, the OS in Q turned seven-dot-oh. Time, as is its wont, passed, and so did I, from back-of-book system-beating columnist to front-of-book eponymous columnist and later from FOB eponymist to BOB Internet traveler, making in all four columns for the same magazine. The more things change, long-time readers must have thought, the more they remain this Swaine. I checked in with *Time* the other day to see if it was done. It wasn't, but it did have some interesting news. The MacOS was still holding at seven-dot-ecks, *Time* said, and was expecting to hold there for a while yet. Version eight was, well, in question. In fact, dead. Trashed. Thrown into the binhex of destiny, *Time* said, by which I understood it meant the dustbin of history. *Time*'s a little dyslexic. Meanwhile, no system nine-dot-oh is revving toward betatude, no bright young understudy from R&D awaits its chance at stardom, no MacOS95 or MacOSNT trembles in the wings, anxious to leap onstage like Booth from the box to cry "Sic semper tyrannis, Bill G_!"

If I have limned this sketch clearly, you should see that as of the precise moment in question (the moment previously identified as "the other day"), Apple had no Macintosh operating-system future. None. This is, I think you'll agree, an interesting position for a computer company to be in. I make a point of clarifying the position because of the remarkable electric shock that went through the entire Mac-observing community when the news came out last fall that Apple was going to replace the current aging operating system with a new one—here comes the good part—written entirely from scratch.

Truthfully, I can't speak for the entire Mac-observing community. An electric shock went through one Mac observer, namely me, on hearing the news.

And it shouldn't have. As an old system beater from way back, I should have been better insulated. This news flash shouldn't have shocked anyone. After all, what options did Apple have? To declare operating systems obsolete? I suppose that thought has crossed the minds of the members of the OpenDoc team. If Steve Jobs were still CEO, that might look like a real option.

However, immediately on the heels of this from-scratch story came the denials. Yes, Doctor Amelio had said something like that, but no, Apple wasn't really going to design and write its next operating system utterly from scratch. Doctor Amelio was just making some general observations about how operating systems are developed. Generally.

### We Could Have a Contest

Apple's flurry of announcements and clarifications left certain details regarding this future MacOS unspecified, such as its features, nature, source, and release year. No doubt Apple will flesh out its plans somewhat someday.

Meanwhile (a technical term from the jargon of publishing, meaning "between the writing of this column in my temporal reality and the reading of it in yours"), it would be silly to try to critique this operating-system-to-be. But it would be fun to offer Apple our suggestions. This is something that we Apple observers do. We Windows watchers (momentarily changing hats here) do it too, but we don't expect Microsoft to listen to

*Michael is editor-at-large for* DDJ. *He can be contacted at mswaine@cruzio.com.*

us. We Apple observers not only expect Apple to listen to us, we expect Apple to do what we say. When the company doesn't follow our advice, we get mad. We are actually Apple Kibitzers.

Anyone can be an Apple Kibitzer. It doesn't cost anything and it's fun.

Care to join in?

Let's tell Apple what we think it should do about its unannounced future operating system. It could even be a contest: Design a new operating system for Apple. (I don't know who'd give out the prizes, though.) I'll start, since I have the talking stick. Apple probably will have left a few details unspecified even in your temporal reality, so you should be able to join in. If not, in the unlikely event that Apple has already shipped the first developer release of a new OS by the time you read this, you can compare my suggestions to the reality.

## Be All That You Can Be

As I write this, January's MacWorld Expo is shaping up to be a real Be-in. BeOS, the operating system, is getting more preshow attention than Apple or any of its third-party partners.

At first glance, this may seem odd. BeOS is the software that runs on the BeBox, the computer announced in 1995 by Be Inc., the company started by former Apple products division president Jean-Louis Gasseé in 1990. The BeBox and BeOS were created to do an end run around existing operating systems and were designed from the ground up to support object-oriented rapid-delivery development in a truly modern operating system. Commendable goals, to be sure.

But the BeBox has hardly taken the world by storm, and most articles about the company and its hardware have tended to use the word "quixotic" to describe Be Inc.'s attempt to stake out its own platform niche in a market that gives at least some indications of having room for only one platform. But BeOS is very interesting to Apple watchers because it is a state-of-the-art operating system, and it runs on PowerPC hardware. In fact, it runs on the Macintosh.

In fact, Apple's hottest licensee, Mac-clone maker Power Computing, has licensed BeOS to offer on its hardware.

In fact, Apple has been negotiating to license BeOS, and rumors have been rampant that:

1. BeOS will be the next MacOS, or
2. Apple will use a licensed version of BeOS to develop its own new (from scratch?) OS the way Microsoft developed the original version of Windows from a Mac system-software license, or
3. something else.

Grafting BeOS onto the elements of system software that Apple would have to keep (if for no other reason than that BeOS doesn't have them) would use up several rolls of masking tape, but it would be a massive PR band-aid for Apple's scarred, battered, and bleeding system-software image. And what BeOS does supply is pretty nifty.

MacOS has what Apple calls "cooperative multitasking." It has a crude, slow virtual-memory system that requires the user to set the size of the VM partition and reboot on changing it. It has no memory protection, and no hardware abstraction layer, so Mac users get to install various system enablers to customize the OS to their hardware. BeOS, on the other hand, has true preemptive multitasking, a modern virtual-memory system, solid memory protection, and a hardware abstraction layer that made it relatively easy for Be to port the OS to Mac hardware. BeOS supports symmetric multiprocessing (on two processors), has a snappy multithreaded user interface, and is fully object oriented, none of which can be said for the current MacOS. Nor, and here's the real point, can it be said for any operating system Apple had ever even planned to ship in the next couple of years. BeOS is only in beta release now, but it is a better operating system than Apple has any hope of shipping in this century without going outside the company.

*The BeBox has hardly taken the world by storm*

So I'd say that Apple could do worse than to embrace BeOS.

Exactly how that would work, both technically and financially, is a question that I will gladly leave to Apple and Be. But if Apple does embrace BeOS, it will be quite a coup for Gasseé and company. Among other things, it will mean that Be Inc. will go from having practically no applications that run on its hardware to inheriting the entire Mac application universe.

Talk about a killer Be app.

## The Quick and the Dead

That's my recommendation, Apple. But if you don't like it, there are other options.

As long as you're considering going outside the company to companies started by former Apple employees, how about General Magic? Their new Rosemary operating system is designed to run on RISC hardware. But then, if you're going to consider a PDA OS, you don't have to leave home: You could just scale up the Newton operating system (or am I supposed to call it "Newton Intelligence?"). Version 2 of the Newton OS is pretty nice. Alone among broadly used operating systems, it eschews the file metaphor, a virtue that could make Newton-based machines accessible to a broader audience than PCs enjoy.

Crazy idea, right? But currently Sun and Microsoft are putting significant efforts into producing, respectively, Java-based operating-system implementations and versions of Windows for smaller and smaller devices. As Microsoft struggles manfully to scale down to PDAs and telephones and watches, maybe you steal a trick by starting with an OS designed for a funky little device to begin with.

Crazy idea, right? But the latest Newton models are testing the boundary between PDA and computer. The Newton 2000 might be everything some people want in a portable computer, for under $1000.00. 162-MHz processor, 1.4 pounds, runs 24 hours on a charge, 480×240 screen with 16 gray levels, and more bundled software than an Osborne 1. Okay, the keyboard is an option, but at least you can get one. And the eMate, designed for school kids, about half the price of a 2000 if purchased in volume (keyboard included), could be all a schoolkid needs in a portable computer. If these machines catch on, maybe Apple should scale the Newton OS up to the Mac line. Or just scrap the Mac line and sell Newtons.

Neither seems likely. Some other scenarios seem neither likely nor appealing. One option would be to graft the Mac GUI on top of NT. That hardly qualifies as a crazy idea, but to those of us who see Apple's only role in the computing universe as being the grit in the bearings of the Microsoft juggernaut, the idea seems unthinkable. I choose not to think about it, anyway.

Of course, inertia always favors continuing with the current plan, or, in the case of Apple today, the current lack of a plan. In this scenario, Apple trickles out minor revs to the OS for the rest of its independent corporate life, charging for them whatever the traffic will bear. If the OS in the Mac is dead, the media level is quick—Quicktime, Quickdraw, and their quick spawn manage to evolve with the times, span platforms, and stay near or at the cutting edge of hot technologies. Too bad you can't turn such system technologies into an operating system. I'd hate to see Apple give us a replay of IBM in 1982: You get to choose your operating system. Fortunately, since they don't have one, they aren't really in a good position to offer choices.

At last spring's World Wide Developer's Conference, new Apple boss Gilbert Amelio, referring to Apple's financial and other crises, predicted that we would wonder a year from now what all the fuss was about. Do you suppose he meant we'd forget all about past disasters in the face of present calamities?

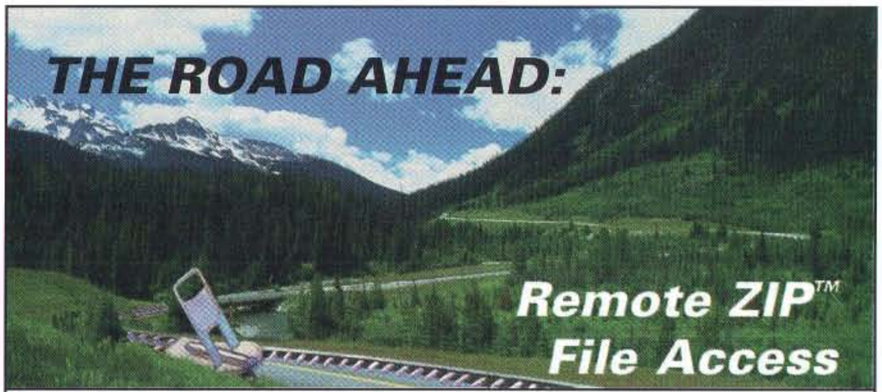
## Java Will Rot Your Teeth

The developer of a HyperCard-like environment for UNIX (I'd better pause for a moment to let you get your mind around that fairly bizarre concept) has brought together what looks like every argument he could think of for why Java is bad, bad, bad. You can read his cogent caveats on the Java hype machine at http://www.csn.net/MetaCard/java.html. Or you can read my summarization, annotation, and augmentation of them here. Some of the following ideas are his, and the rest came to me in a dream. Here are ten reasons why Java is bad for you:

1. We don't need another C++. One is bad enough.
2. It's too stripped down for many programming purposes, the legacy of being designed originally to run on smart toasters or something. Needed or desired features may be added on later, but having the right features designed in from the start is the way to go.
3. Multithreading is a pretty obscure feature to include considering the things left out.
4. Multithreading will rot your teeth.
5. Java is slow.
6. It'll never be fast. Without explicit memory management and the use of pointers, Java can never match C for fast executable code. And it's interpreted, so the user has to wait for the interpretation phase to see the application or applet execute. Just-in-time compilation doesn't help.
7. Most of Java's alleged virtues (portability, ability to deliver functionality over the Internet, ability to screen code for security purposes) come from its being interpreted. Other interpreted languages have or could easily have these same virtues and are at the same time easier to program in than Java. Some interpreted languages also have large libraries to draw upon to speed development. We don't need another Basic.
8. The development environments currently available leave something to be desired.
9. How open a standard is it when a runtime environment license costs $125,000? And when Sun controls the development of the language? And licensees are required to turn any improvements they make over to Sun?
10. In the late 1990s, what we need are more and higher high-level languages that can speed development of more powerful, media-rich applications and that can call upon compiled routines written in an efficient low-level language when they need to. We don't need another C.

That's the list. Presented for your perusal in the spirit of devil's advocacy, and not necessarily reflecting the views of the management. Personally, I like Java. But I do believe it might be just a tiny bit overhyped.

**DDJ**



**Open ZIP Files Across a Network with NetZIP™ Remote...**

Give your ZIP program extra capabilities by adding an on-line feature using the NetZIP Remote tool kit. Clients can now open a ZIP file without having to download the whole file. Now servers can eliminate unnecessary download traffic and increase bandwidth, and clients save time by downloading the right file the first time.

Attention Developers of
- Windows™ Networks
- Company Intranets
- FTP Client Software
- Compression Software

Whether you use C, C++, or Java to write your network applications, you can use the NetZIP tool kit to give any existing Windows ZIP engine a remote Windows interface.

NetZIP Remote Products:
- Windows Client Tool Kit
- Java Client Tool Kit
- NetZIP Server

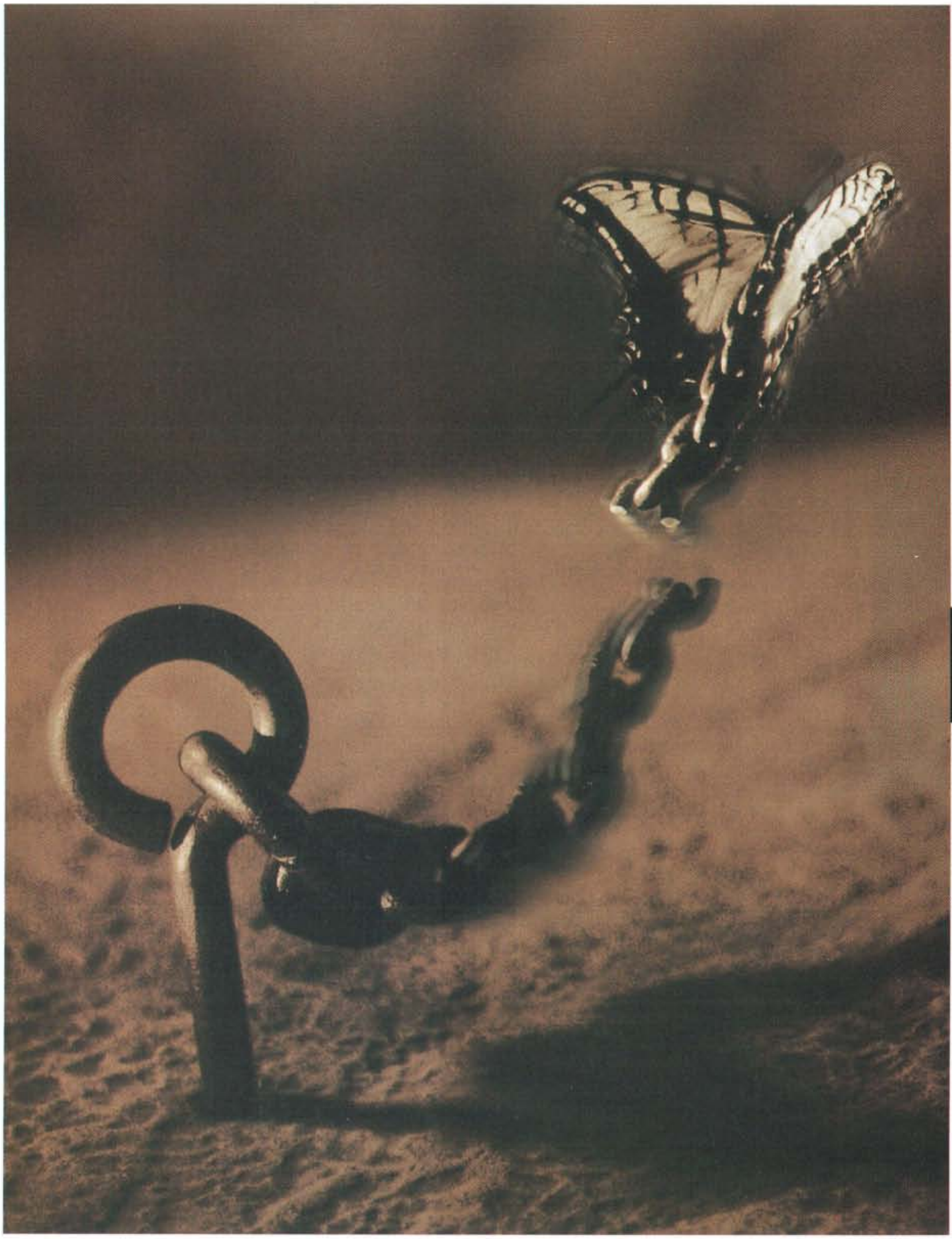
**Software Builders Products**
**http://www.netzip.com/**

| | |
|---|---|
| Sales: | 800-432-0025 |
| Fax: | 770-395-1616 |
| International: | 770-395-6465 |

**400 Perimeter Center Terrace North, Suite 445, Atlanta, GA 30346, USA**

Windows is a trademark of Microsoft Corporation. Remote ZIP and NetZIP are trademarks of Software Builders International, Inc.

CIRCLE NO. 333 ON READER SERVICE CARD



PENTIUM® PRO
PROCESSOR

With all due respect to RISC/UNIX-based systems, we think you'll find the Compaq Professional Workstation offers something that's been sorely missing in proprietary workstations. Namely, freedom.

To begin with, you'll have plenty of power to run your specialized applications. This is made possible through a range of cutting-edge performance features. Including Compaq's advanced system architecture which is optimized for Windows® NT and can run up to two Pentium® Pro processors. And because our workstation is based on open systems standards, you'll find it will integrate easily into your existing network. So instead of having to work within the constraints of a proprietary system, you'll have the flexibility to accommodate your needs, whatever they are. Of course, with Distributed Access, you'll also be assured of a transparent connection to all the information you need throughout your enterprise. Even in RISC/UNIX environments.

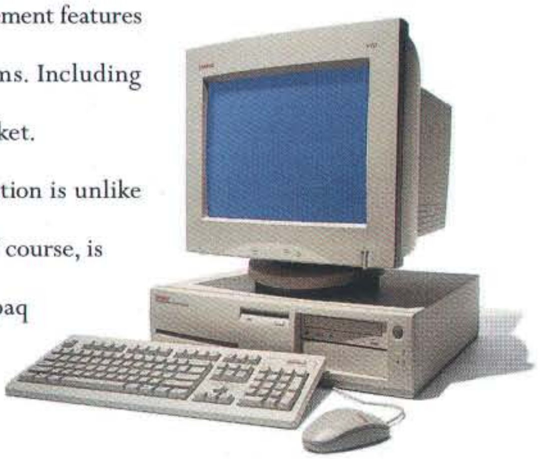
distributed @ access

Another benefit is the result of our partnerships with leading independent software vendors like Microsoft, SDRC, Autodesk and PTC. Because these solutions have been thoroughly tested, you'll get optimum performance and compatibility.

## YOUR RELIANCE ON CONVENTIONAL WORKSTATIONS IS ABOUT TO CHANGE FOREVER.

Finally, our workstation provides a lower cost of ownership—not only through price:performance but also through Compaq's industry-leading management features and comprehensive service and support programs. Including hundreds of resellers specially trained for your market.

All said, the Compaq Professional Workstation is unlike any workstation you've ever used before. Which, of course, is exactly the point. For more information on Compaq workstations or Distributed Access, visit us at www.compaq.com or call 1-800-318-7774.



*So what's under the hood? 1–2 200MHz Pentium® Pro processors with NT 4.0, a 256K cache, up to 512MB of ECC DIMM memory, an Ultra-Wide SCSI controller, and advanced 2D/3D graphics accelerators.*

COMPAQ

Has It Changed Your Life Yet?

CIRCLE NO. 329 ON READER SERVICE CARD

*Dr. Dobb's*

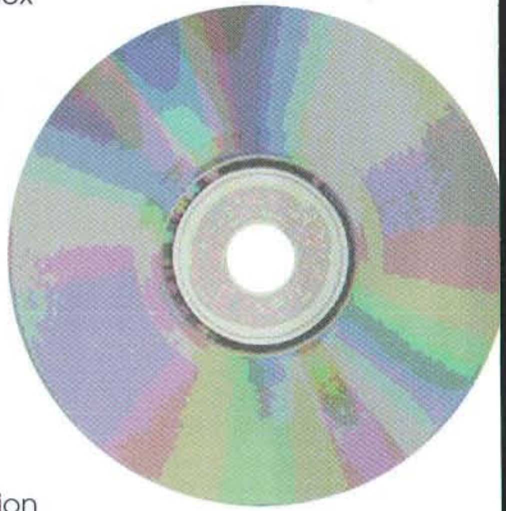
# Distributed Objects Development CD-ROM

New Release

Dr. Dobb's CD-ROM LIBRARY

The latest in distributed object technology, including languages, libraries, and environments

Programming distributed systems using traditional tools is both complex and error prone. That's why the editors of *Dr. Dobb's* assembled this collection. Using this CD-ROM will enable you to build reliable distributed, object-oriented applications. And since the Internet and World Wide Web are rapidly increasing the demand for distributed applications, these tools and utilities are at the very heart of this technology.



This CD-ROM contains over 100 megabytes, featuring:

- **Arjuna 3.3.2**—A robust O-O tool for building fault-tolerant distributed applications out of C++ programs
- **Shadows** —Creates World Wide Web Arjuna applications
- **Interlanguage Unification System (ILU)**—A powerful programming model for distributing applications via an Interface Description
- **Obliq**—An O-O scripting language supporting distributed computation
- **Visual Obliq**—The World Wide Web version of Obliq
- **Phantom** —A stand-alone, interpreted language for developing large-scale, interactive, distributed applications
- **Distributed Python**—A portable, interpreted, O-O language which is extendable
- **Hector**—A framework for providing open negotiations and interoperability for applications

For a detailed description, please visit our web site: **www.ddj.com/cdrom/**

Order your copy today!

YES Please send my CD-ROM today. I understand that I have a full money-back guarantee.

**Distributed Objects Development CD-ROM**......**$39.99**

Please add sales tax in the following states: CA (8.5%), GA (6%), IL (6.25%), KS (6.9%), NY (8.25%), TX (8.25%), Canada (7% GST). $____

Shipping ($2.00 for each CD-ROM shipped in USA/Canada, $10.00 for Airmail to all other countries.) $____

**TOTAL AMOUNT**...... $____

Name

Address

City/State/Zip

Country

Phone E-mail

Here's how I'm paying (circle one): VISA - MC - AMEX - Check enclosed

Credit card number (include all digits) Expiration Date

Signature required S197

Please allow 2 weeks for delivery in the U.S./Canada; 4 weeks for all other countries, Checks/money orders must be drawn on a U.S. funds only.

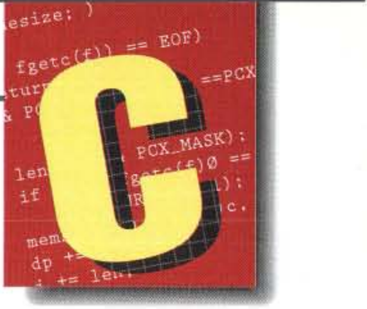
CALL TOLL FREE 800-500-6841

**INT'L Orders:**
Use mail, fax, e-mail, or call 913-841-1631

**FAX: 913-841-2624**

**E-Mail Orders:**
orders@mfi.com

**Mail Orders:**
Dr. Dobb's CD-ROM Library
1601 West 23rd St., Ste. 200
Lawrence, KS 66046-2700 USA



# Space Shuttles, Tomato Cans, and Teenage Daughters

Al Stevens

The theme of this month's issue is testing, so I'll devote the first part of this column to a reminiscence of my participation in a large test project.

During the Apollo program, I was a certified ACE operator. ACE, short for "Apollo Checkout Equipment," was a configuration of consoles, computers, sensors, and transducers NASA used to test space capsules on the launch pad. After a one-week class, I was qualified to operate an ACE console. In movies about space exploration, when the astronauts talk to Mission Control, the guys in Houston always have crew cuts, sit at consoles, wear white shirts and ties, chain-smoke cigarettes, and look either cool or worried depending on what's happening. I was not one of those guys. Mission Control is in Houston. I worked at Kennedy Space Center; we tested the vehicles and lit the fuses, but Houston ran the show—a gift from Lyndon Johnson to his constituents.

Despite my certification, I never got to sit at an ACE console. They sent me to school to learn the system so that I could write programs to support it. Here's how it worked. Imagine a console like the ones that you see in the movies. Imagine me 30 years younger with a crew cut, sitting at the console. Smoking a cigarette. Looking cool. Because nothing ever goes wrong on my watch. Or when it does, I'm still cool.

The console has buttons and dials for input, and gauges and lights for output. The input and output devices are connected to a computer. The computer can sense when I press a button or change a dial's setting. The computer can turn the console's indicator lights on and off, and position the needles in the gauges.

Now imagine an array of cables that hang out the back of the computer. They are connected to D/A and A/D converters in the computer. The other ends are connected to sensors and transducers that can themselves be connected to something—typically a space capsule or launch vehicle—that is to be stimulated and tested. The computer reads the digital values converted from the sensors' analog data and writes digital values to be converted into analog values for the transducers to emit.

I am imagining this configuration along with you because they never let me actually see it. You had to look like Ed Harris or Gregory Peck to be allowed in the room with the computer and consoles. I was more the Rick Moranis, Wally Cox type. (Hunk/nerd actors selected from two generations so that readers from both generations can relate.)

In the space program, every launch is unique. Space exploration is an ongoing R&D activity, a never-ending management of crises, so the configuration of every vehicle is different in one way or another from those that precede it. Consequently, the procedures for testing a vehicle had to be custom-designed and custom-built for each particular launch. A complex table of parameters told the ACE computer which transducer values to change when the ACE operator pressed a console button or changed a dial and what console indicator to change based on the values read by the sensors. The sensors and transducers were connected to the vehicle at appropriate test points. A crew-cut person called the Test Conductor directed everything from a test procedure script while the crew-cut ACE operators punched the buttons, turned the dials, and read and reported the results. My role was to write programs on a different computer (which I was allowed to see) to generate the test procedure scripts for the Test Conductor and the test parameters for the ACE computer.

Pure oxygen and sparks don't mix. In January 1967, three astronauts perished in the infamous Apollo fire during such a test, and new occupations resulted in the space program. Spin doctors. Blame shifters. Fault deflectors. The darkest six-month period in my professional memory followed, and I left the space program not to return for 13 years.

History repeated itself in January 1986 with the Challenger accident. I wasn't involved with the vehicles anymore, but I watched every launch. Still do. It is interesting to observe one significant difference between those two moments in history. A cultural difference. After the Apollo fire, there were no jokes. But something happened to the American sense of humor in the interim. Somewhere between Gregory Peck and Rick Moranis, we turned to humor as a means of dealing with the emotionally unacceptable. Today, every national tragedy is followed almost immediately by a spate of jokes. Take it from me, you can't watch seven people get blown to Kingdom Come without being changed, and you can't see anything funny about it.

NASA still tests everything, of course, and they look for innovative, efficient ways to solve the unique problems of testing things that are headed for space. In the 1980s, the emphasis turned to Shuttle payloads. A Space Shuttle carries cargo—its payloads—into space. That's its purpose, and, typical of space exploration,

*Al is a* DDJ *contributing editor. He can be contacted at 71101.1262@compuserve.com.*

every launch has a unique payload configuration. Ultimately, if budgets return, the Shuttle will be used to carry pieces of the Space Station to where astronauts can float around and assemble the Station like a huge orbital erector-set project. All those parts and pieces will be manufactured by different companies in different locations. Specific components will join one another in a Shuttle's cargo bay to be ferried into orbit. It costs a lot of money to put something into orbit. Anything that goes up needs to be tested thoroughly to ensure that we're not wasting fuel and rockets and time by taking junk into space. Every payload component needs to be tested at the factory, in the payload processing facilities, on the launch pad, and in orbit.

One of NASA's goals was to extend the old ACE concept so that, rather than use a file of cryptic parameters, a computer could run an interpreted language that expressed the test sequence in procedural program code. They called this concept the Space Station Operational Language (SSOL). The idea was that a common language could be hosted by different platforms to run the same program to test a component at different locations with the same results. It would have the control constructs of a structured programming language with the ability to directly address hardware registers to control the I/O devices. Sound familiar? Sounds like C. Furthermore, the language had to be intuitive so that engineers, installers, and astronauts who do not write programs all the time could still use it. Nope. Does not sound like C.

I got involved because of my C background and because, true or not, C was widely touted as the only portable programming language extant. C was a likely candidate, not to be SSOL, but to implement SSOL to run on various platforms. C++ was not well known outside of AT&T at the time, but based on what I had read about SmallTalk, I was pushing for an object-oriented approach for SSOL, an interpretable extension to a C-like syntax without the C gotchas that trip up even veteran programmers from time to time. Nobody really knew what I was talking about. Neither did I at the time. As it turns out, I was talking about Java.

The SSOL project eventually faded into obscurity as budget cuts took their toll on the Space Station program. (SSOL became, so to speak, S.O.L.) The results of the study were filed for later resurrection should the money ever start flowing again. When that happens, when we decide again to fund and follow our natural impulse to explore new worlds (after we balance the budget, reduce the deficit, reform campaign financing, eliminate war and world hunger, save Social Security and Medicare, sanction same-sex marriage, and forget about O.J. Simpson), the SSOL project, or something like it, will be resuscitated because the problems remain to be solved.

One of the obstacles in that project was our lack of, not vision, but forward-looking visibility. NASA planned the Space Station to be a 30-year program. We were charged with defining a programming language that would remain relevant for that duration. But when we considered how programming had changed in the prior 30 years, it seemed impossible that anyone could predict in 1987 what programming would be like in 2017. Now, in 1997, the state of programming has already changed in ways that were totally unforeseen then. Given the recent research being made in visual programming, I'd love to get another crack at that language.

*MIDI Xchg stresses the MIDI data stream by sending event messages from one device to another*

**MIDI Potential**

ACE was a huge process-control system applied to solve a test-bed problem. Process control conjures images of many things. Robotics. Real time. Embedded systems. I always imagine a conveyor belt moving tin cans through the factory. One part of the process dumps tomatoes into the cans. A second stage seals the cans. A third pastes labels on the cans, and so on.

That rendition suggests the traditional, pre-1980s fear that computers are replacing workers. Years ago, when someone asked what I did for a living, I would say that I was a computer programmer. That response was inevitably met with a litany of complaints about how computers were putting people out of work. Eventually, I learned to keep my mouth shut. Years later, when computers entered the mainstream and people became accustomed to having them around, I relaxed and once again began admitting my profession when asked. Now my response is met with a detailed description of the other person's latest PC acquisition and a request that I drop by one day next week for a drink and, oh by the way, maybe I could help install Windows 95. I've got to learn to keep my mouth shut.

A process-control computer senses and commands the components of an assembly line. A well-formed process-control system is as generic as the ACE system. It senses generic events and knows how to command generic control devices. A table of parameters—a program—tells the system what those devices are and how to enact the procedures of the particular process.

Recently, I watched such a process being demonstrated on a TV program about the technology of making movies. This particular segment was about sound effects. The sound effects technician (artist, really) sat at two keyboards—an electronic 88-key piano keyboard and a PC keyboard. His 21-inch computer monitor displayed what was obviously a sequencer program, such as CakeWalk or WinJammer, running under Windows 95. A second video monitor displayed the action of the movie. The technician played notes—pressed keys—on the electronic keyboard and his sound system generated sound effects. Not musical notes, mind you, but atonal, eerie sounds. He used combinations of notes to generate combinations of sounds to create the spooky sound effects required by the movie he was supporting.

This piqued my interest. While developing MidiFitz (a real-time rhythm accompaniment program that I published in this column last year), I gained an interest in the Musical Instrument Digital Interface (MIDI), which is the underlying technology that drives electronic music. That TV documentary was about an application of MIDI not related to conventional music at all. (Unless you, like Dracula, think that sounds such as the howling of wolves are music.) A visit to the local newsstand revealed several magazines devoted to electronic music and musical productions. Notable among them is *Keyboard* magazine, another Miller Freeman publication. There are several others. They feature articles mainly about the technology, but include a lot of photos and information about the performers as well. Like the crew cuts of 30 years ago, these guys all tend to look alike. They frown a lot, eschew shirts with sleeves, and sport earrings, body piercing, and tattoos. You

“What makes Velocis the *fastest* database?”

NEW!
High Performance
REPORT WRITER
Available

You do! Velocis™ Database Server gives you speed-enhancing architectural choices and APIs, including SQL C-API, low-level C-API, C++ class libraries, and support for custom APIs. Typical relational client/server database products just can't keep up.

And there's more! Velocis supports both relational and pointer-based network model databases in any combination as well as processing on either side of the client/server equation.

The choice s of multiple operating platforms, APIs, processing localities (client or server), and database models can be combined to satisfy the performance requirements of virtually any application, regardless of database size or complexity.

**Call 1-800-ASK-4-RAIMA**

*If it's fast – it's*
RAIMA™

For more information on what makes Velocis the fastest Database, call for a copy of the *Bill of Materials Benchmark.*

| Velocis | Typical RDBMS | Features |
|---|---|---|
| ✔ | ✔ | ANSI SQL (DML, DDL, DCL) |
| ✔ | ✔ | Client/Server Architecture |
| ✔ | ✔ | RPC Mechanism |
| ✔ | ✔ | SQL C-API |
| ✔ | | Administrative SQL C-API |
| ✔ | | Low-Level C-API |
| ✔ | | C++ Class Libraries |
| ✔ | | Custom API |
| ✔ | | Both SQL & C Stored Procedures |
| ✔ | | C-based Triggers |
| ✔ | | User Defined Functions |
| ✔ | | Server Extensions |
| ✔ | ✔ | Relational Model |
| ✔ | | Pointer-based Network Model |
| ✔ | | Combined Database Model |
| ✔ | ✔ | Primary/Foreign Keys |
| ✔ | ✔ | Referential Integrity |
| ✔ | ✔ | Transaction Processing |
| ✔ | ✔ | Transaction Protection |
| ✔ | ✔ | Roll Forward Recovery |
| ✔ | | Asynchronous Transactions |

**Test drive Velocis Database Server today at www.raima.com.**

**Raima Corporation**, 1605 NW Sammamish Rd. #200, Issaquah, WA 98027 USA
Inside the U.S.: 1-800-275-4724, Outside the U.S.: 206-557-0200, Fax: 206-557-5200, Web: www.raima.com, Internet: sales@raima.com

**Germany:** 49 7022 9256 10 **Singapore:** 65 334 0061 **Italy:** 39 49 72 05 77 **Argentina:** 54 1 470 72 03 **Spain:** 34 3 225 39 95
**Raima UK Ltd.:** 44 1 273 819 292 **Australia:** 61 2 419 7177 **Denmark:** 45 44 88 99 00 **Raima Benelux:** 31 35 694 4738 **Colombia:** 57 1 62 10 070
**France:** 33 1 446 100 42 **Sweden:** 46 13 311 588 **Baltic States:** 372 631 4009 **Finland:** 358 0 8045 130 **Japan:** 81 3 5548 6926 **Russia:** 7 812 316 1965

*(continued from page 98)*
might think they are cool now, but wait until one of them is at your door to take your teenage daughter to the prom.

What I learned from these magazines comes as no surprise to the cognoscenti, that MIDI is the process-control technology that drives not only the drum machine, but the lights, smoke, lasers, and anything else electro-mechanical associated with a rock concert extravaganza. My teenage daughter once played one of her albums for me and went on and on about the production effects at the concert she attended. In the finale, she said, the band went berserk and smashed all their instruments on the stage. I listened to the album and opined that they had it backwards; they should've smashed those instruments before they were allowed to play them. She responded with a "Fawther!" and left the room.

The sound-effects tech on the TV documentary had combined several MIDI and PC components to construct a sound-effects workbench. He used waveform tools to build the individual sounds. Then he loaded these sound files into a musical sampler, which is normally used to store the sampled recordings of real instrument notes. He assigned the samples as patches (voices) of the virtual MIDI instruments. Then, when he played notes on the keyboard, sound effects came out instead of music.

### MIDI Fundamentals

MIDI is an electrical specification and a protocol. The most remarkable thing about MIDI is that in an unprecedented spirit of cooperation, an entire industry embraced the MIDI standard without being forced to by market pressure.

When you press a key on a MIDI keyboard, the keyboard sends a three-byte serial data packet to its MIDI Out port. The keyboard might also generate audio sounds to synthesize the notes of an organ or an acoustic piano, but that action is unrelated to the MIDI scenario. The first byte of the packet identifies the event and a four-bit channel number. The event is, in this case, the Note On message. The second byte identifies which note you pressed. Keyboards have 88 keys, and their notes are assigned the values 21 to 108. The third byte of the MIDI-event message packet contains a value called the velocity, which numerically represents how hard you pressed the key when you played it.

When you release the key, the keyboard sends a corresponding Note Off message. The piano's controller devices, such as the sustain pedal and pitch blend wheel, send controller messages when you use them.

MIDI includes other messages to signal the start and stop of a performance, clock synchronizing events, and so on.

You can connect a keyboard's MIDI Out port to any device that has a MIDI In port. Many keyboards have a MIDI In port. If you connect two keyboards, the notes you play on one are heard through the audio playback of the other.

Coordination of all these messages into a performance is the job of a sequencer device, which is usually a program running on a PC or Mac. All the instruments are connected to the sequencer in a daisy chain or in a star network. The sequencer operator (a person) assigns each instrument device to one of the 16 channels, and the sequencer program assigns one of the standard MIDI voices to each of the instrument channels. There are 128 standard voices called "patches," and they are organized into instrument groups with one set of voices for pianos, another for strings, and so on. The drum machine has its own set of patches.

*Electronic musicians have come to know and work with the limitations of the MIDI bandwidth*

Sequencer programmers (people) use the sequencer program to build Standard MIDI Format (SMF) files containing tracks of MIDI events, with each track assigned to a channel. The SMF file also records data related to the overall production, such as time signature and tempo. (Contemporary sequencer programs can also mix audio in with the MIDI events, but that technology is outside the MIDI standard and involves proprietary file formats.)

During playback, the sequencer program reads the SMF file and sends the MIDI event messages to the MIDI devices via the sequencer's MIDI Out port. Each device reacts only to the events assigned to the device's channel, which is assigned by the system operator (one of the musicians, usually) when he or she sets up the system. The MIDI system plays the production, and the live musicians jump around the stage and play along, adding the human element to the performance and exciting your teenage daughter to dangerous levels. If it is done well, the audience can imagine that a 30-piece orchestra and several lighting and effects technicians are hidden away backstage.

So, the studio and the stage and the factory have a lot in common. It is told that a musician showed up for a recording date to find 30 other musicians setting up in the studio for the session. The newly arrived musician looked around and said, "I hope you people realize that you are putting three sequencer operators out of work."

### MIDI versus Process Control

Back to the factory. Consider the cost of process-control peripherals and their interfaces in the computer. Might there be something already in place that could do the same job? The Mac has MIDI connectors built in. Every PC with a SoundBlaster or compatible sound card has MIDI In and Out hardware ports. All you need is an adapter cable that takes the signals from the game port adapter to corresponding MIDI In and Out five-pin DIN connectors. Turtle Beach includes such a cable with its sound cards. The Sound Blaster MIDI Interface Adapter is an extra-cost option. These adapters are not just cables, by the way. There are some built-in electronic components.

Windows 3.1/95/NT have built-in MIDI drivers and applets. Sequencer programs are readily available.

Suppose that you built your factory devices or your payload-testing devices or the auto-animatronic figures at your theme park with MIDI interface hardware. You could orchestrate an assembly line or a widget test or the Gettysburg Address with an electronic keyboard and a sequencer.

What are the advantages of using MIDI input/output hardware for process controllers instead of other standard interfaces, such as the serial port? The main advantage is software. The drivers are built into the operating system. Sequencers are plentiful and inexpensive. The SMF file format is standard. You can play your sequence through the Media Player applet that comes with Windows. You can simulate input devices with a standard MIDI keyboard.

What are the disadvantages? One might be bandwidth limitations. Another might be the absence of error detection and correction.

Electronic musicians have come to know and work with the limitations of the MIDI bandwidth. The speed of the sequencer computer is a factor. The timing of MIDI messages is a function of the tempo assigned to the performance, which essentially throttles the data rate. The bandwidth to support that data rate is a function of how many devices are in the network, how many notes typically need to be played on those devices simultaneously, and whether the network topology is a daisy chain or a star.

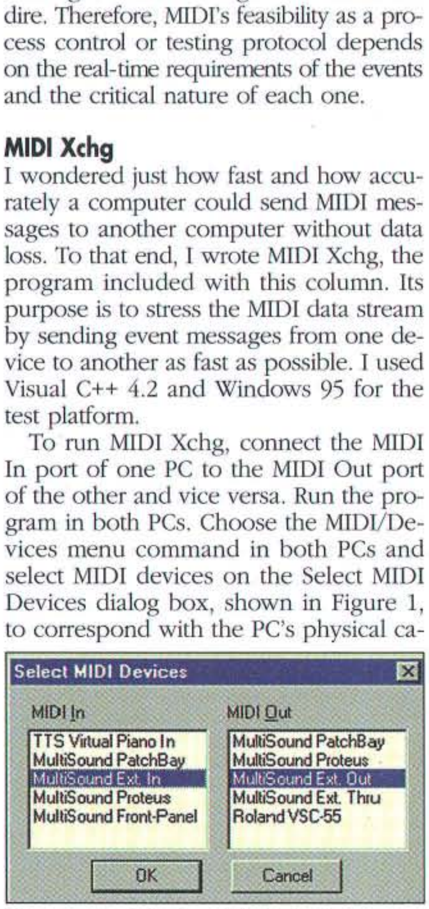
MIDI is a real-time protocol, and, as such, it has no built-in error detection and correction. It's a lossy protocol in that it doesn't really matter if an occasional event gets lost. If you miss a cello note from somewhere deep inside a symphony, no one notices, or, if they do, the consequences are only aesthetic—assuming that they are not employed as music critics. There are no ACKs and NAKs, no CRCs or checksums, and no retransmitted messages. You might stop and replay a passage when you are practicing, but on-stage you stay cool, pretend the mistake didn't happen, and just keep playing.

You can miss a wolf howl, cello note, or tomato can, or your autoanimatronic Abe Lincoln can skip a gesture, but the consequences of missing a critical voltage reading or hatch configuration could be dire. Therefore, MIDI's feasibility as a process control or testing protocol depends on the real-time requirements of the events and the critical nature of each one.

## MIDI Xchg

I wondered just how fast and how accurately a computer could send MIDI messages to another computer without data loss. To that end, I wrote MIDI Xchg, the program included with this column. Its purpose is to stress the MIDI data stream by sending event messages from one device to another as fast as possible. I used Visual C++ 4.2 and Windows 95 for the test platform.

To run MIDI Xchg, connect the MIDI In port of one PC to the MIDI Out port of the other and vice versa. Run the program in both PCs. Choose the MIDI/Devices menu command in both PCs and select MIDI devices on the Select MIDI Devices dialog box, shown in Figure 1, to correspond with the PC's physical capable ports rather than the sound card synthesizers. Choose the Receive command on the MIDI menu of one of the PCs. Then choose the MIDI/Send Data command on the other. The sending PC begins sending Note On messages to its MIDI Out port. The data stream consists of repeated loops of 128 notes in the sequence of 0 to 127 so that the receiving PC can determine if any notes are missing in the stream. Wait a few seconds and choose the MIDI/Stop Sending command to stop the data stream. The receiving computer reports the number of events received, the number of notes dropped, the number of events per second, and the approximate effective baud rate just to provide a familiar measure with which to compare the performance. Figure 2 shows the application window after a session.



| MIDI In | MIDI Out |
|---|---|
| TTS Virtual Piano In | MultiSound PatchBay |
| MultiSound PatchBay | MultiSound Proteus |
| MultiSound Ext. In | MultiSound Ext. Out |
| MultiSound Proteus | MultiSound Ext. Thru |
| MultiSound Front-Panel | Roland VSC-55 |

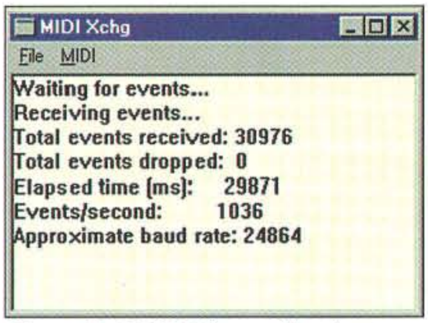
***Figure 1:** Selecting MIDI devices.*

## The MIDI Xchg Implementation

MIDI Xchg is an unremarkable MFC application with no document/view architecture but with application and frame classes. There are ten source-code files and a resource file. The entire project is available electronically; see "Availability," page 3. The application and frame classes each have headers and .cpp files. I adapted the *MidiIn* (see Listings One and Two; listings begin on page 113) and *MidiOut* classes (Listings Three and Four) from MidiFitz for this project. Those classes each have a header and a .cpp file, and those are the classes I'll discuss here. The two classes encapsulate enough of the Windows MIDI API to support this test. When the program instantiates the two objects, their constructors call the *midiOutGetNumDevs* and *midiInGetNumDevs* API functions to determine the number of MIDI devices available. Both classes have a *DeviceList* member function that loads a *CListBox* object, passed as a reference argument, with the names of the installed devices. MIDI Xchg's *Select* MIDI Devices dialog box uses these functions.

When you select MIDI devices, the program calls the *ChangeDevice* member function, which closes the current devices and opens the new ones. The *midiInOpen* API function has a parameter that is the address of a function for the system to call when a MIDI in message is received. The *MidiIn* class includes a *RegisterMIDIFunction* member function so that the program can provide the address of its MIDI-message input function. This function must be called ahead of *ChangeDevice* so that the *MidiIn* object has a function address to provide to *midiInOpen* from *ChangeDevice*.

When the sending PC begins sending event messages, it first sends the MIDI start event message by calling *MidiOut::StartMessage*. When you choose the Stop Sending command, the program calls *MidiOut::StopMessage* to send the MIDI stop message. The receiving program uses these two messages to time the data flow.



```
Waiting for events...
Receiving events...
Total events received: 30976
Total events dropped: 0
Elapsed time (ms):    29871
Events/second:        1036
Approximate baud rate: 24864
```

***Figure 2:** MIDI Xchg application window.*

## Conclusion

While working on this program, I learned several things about the way that the Win32 API handles MIDI. First, it buffers MIDI input messages so that if the program does not get around to processing them right away, they come in later. I was using the callback window option rather than the callback function for MIDI input messages so that I could post their reception in the program's main frame client window. You can't call most system functions from a callback function. The latent Win32 overhead for receiving the messages and displaying them was high enough that the messages backed up in the receiver. When I stopped the transmission, there was a burst of backed-up messages displayed, and the last bunch of them had a lot of missing messages.

By changing to a callback function and using that procedure to count the messages received and the messages dropped, I eliminated the bottleneck, and the MIDI Xchg receiver was able to receive and process as many messages as the MIDI Xchg sender could send.

The sender sends the messages in a tight loop by calling the *MidiOut::SendEvent* function with the Note on messages already formatted. That function calls the API's *midiOutShortMsg* message. It became apparent from the results that *midiOutShortMsg* does not return until the message has been sent, or, at least, waits until the previous message has been sent and the output port is available. That strategy effectively enforces the MIDI data rate, which turns out to be something close to 28K baud, about 1040 events per second, which ought to be plenty fast enough to paste labels on tomato cans.

**DDJ**

**(Listings begin on page 113.)**

Dr. Dobb's

## World Wide Web Toolkit CD-ROM

## $29.95

The *World Wide Web Toolkit CD-ROM* is the most comprehensive compendium of software resources relating to all aspects of the World Wide Web. Included on this CD-ROM:

- Web Browsers
- HTML Authoring Tools
- Search Engines
- Web Servers
- Conversion Utilities
- Plus Much More!

Contains tools for all the major platforms used for the Web today, including Unix, 32-bit Windows, 16-bit Windows, Macintosh and OS/2. Many of these tools come with complete source code for C, C++ or Perl.

Save time and trouble–avoid hours of costly downloading! Browse the CD-ROM using your own Web browser. Follow URL pointers on the CD to download any tool on the CD.

**Call: U.S. & Canada: 800-992-0549**
**All other countries: 913-841-1631**
**Go to our website for more product information:**
**www.ddj.com/cdrom/**

**Fax Orders:** 913-841-2624

**E-Mail Orders:** orders@mfi.com

**Mail Orders:** Dr. Dobb's CD-ROM Library
1601 West 23rd St., Suite 200, Lawrence, KS 66046-2700 USA

NuMega™ Technologies

## Change Your View... Be A Lead Dog!

***Software Engineering***

Are you ready to lead the pack? We are expanding the NuMega team to create the next wave of award-winning tools for Windows 95 and NT. If you're one of the best and need to be on the cutting edge of technology, consider one of the following opportunities.

**Windows GUI Expert**
proven experience in C/C++, MFC, and UI design and development methodologies is required. Knowledge of Windows internals a big plus.

**Windows Internals expert**
You'll need extensive knowledge of Intel x86 architecture, Windows internals.VxDs,and system-level code for this position. A proven system development track record in Assembler, C, and C++ required.

**Software Programmers**
Apprentice positions available for those with long-term goals of mastering their craft. Candidates should have strong communication skills and experience with PCs, Assembler, C, C++, Windows, and DOS

email your resume to jobs@numega.com, or send it to:
NuMega Technologies
9 Townsend West
Nashua, NH 03063
Att: Human Resources

# You're the best. We're the best. Let's get together.

CMG is a medium large software house, based in the Netherlands, the UK and Germany. CMG is well on its way becoming one of the largest independent IT companies in Europe.

The Advanced Technology division in the Netherlands is expanding rapidly and, as a consequence we are looking for more consultants.

We need the best project managers, the best technical project leaders, the best business consultants, the best infrastructure specialists and the best software engineers. In brief: we are looking for you. If you are a (Dutch-speaking) professional, then join the other 3,000 CMG people who are currently helping our customers expand their business, solve IT problems and are making a great career for themselves.

We work hard, we play hard and we pay above market-rates. In a nutshell, this is the most accurate way to describe how we succeeded as a profitable independent company for over 30 years in the fastest growing marketplace in the world.

You're the best and we're the best to work for, according to an independent market analyst who publishes a yearly report called 'The best 25 IT-companies to work for'. In this list, CMG is mostly ranked first or second. Are you interested in what we do? Then visit our website, and form your own opinion.

Do you live in the Netherlands or are you planning to do so in the near future? Let us know and send your CV by mail, E-mail, fax or snail-mail, and we'll take it from there. Of course, you can also call us or use the recruitment form on the CMG homepage. CMG Finance, division Advanced Technology, P.O. Box 133, 1180 AC Amstelveen, The Netherlands. Rob Labordus, Associate Director.

Phone ++31 20 50 33 000, Fax ++31 20 50 33 011, E-mail rob.labordus@cmg.nl CMG's Internet address: *http://www.cmg.nl*

CMG
Information Technology

# How Do I Access a SQL Database from an Applet?

Cliff Berg

In the past year, a lot has been written about the usefulness of the Web as an alternative to conventional custom software-based solutions for distributing business and other information. In the traditional approach, a vertical application is programmed using embedded SQL or some other technique, and application programs that access the database are distributed to users within a corporation. One disadvantage of this approach is that the application has to be redistributed every time it is changed. The web-based alternative instead relies on the simple (but powerful) HTML publishing model for the user interface and on centrally maintained CGI programs for accessing the database and making the data available to web pages.

While this approach is an improvement in cost and flexibility over the deployed application approach— and much has been written about the cost advantages of doing things this way— it suffers from the very limited feature set provided by the CGI protocol, which is primarily a transaction-oriented mechanism in which a server program accepts a parameter stream from the user's web-page form, then generates a new web page as output, and possibly accesses a database along the way. Tools such as Javascript make it possible to perform local processing prior to sending the query to the database— for example, to do checking on fields— but Javascript is a poor choice for large applications. Because it does not provide a mechanism for reusing code, commonly used routines must be cut-and-pasted into each place they are needed.

A better approach is to utilize the distributed programming model of Java, so that we can have the best of both worlds: a flexible and highly interactive (even real-time) UI, but with the low cost of a web-based approach. The need to use CGI, which is notoriously difficult to debug, is also obviated.

*Cliff, vice president of technology of Digital Focus, can be contacted at cliffbdf@digitalfocus.com. To submit questions, check out the Java Developer FAQ Web site at http://www.digitalfocus.com/faq/.*

## JDBC

Java 1.1 will incorporate a package called "java.sql," which provides a standard Java binding for accessing an ODBC-compliant database. This package is available now as a beta release from Sun's Java site (http://java.sun.com/). To use it, all you need is a JDBC/ODBC-compliant driver. Eventually, such drivers will be available in Java-only versions. At the moment, most are implemented with Java native methods which, in turn, call ODBC driver libraries on a platform-specific basis. Since native methods cannot be called from within a remotely loaded applet in most browsers, this limits these drivers to applications that run on the same host as the database. This is not practical for a web application.

The solution implemented by many driver providers is to use a special server program that resides on the same machine as the database and fields remote JDBC requests, then calls the required ODBC driver functions. To achieve this, the JDBC driver (which runs within the applet) must be designed to communicate with this special server program.

In this column, I'll use a driver provided by XDB Systems (available from http://www.xdb.com/). All of the code will use the standard JDBC packages. The only vendor-specific impact on the program is the instantiation of XDB's driver. Most vendors, including XDB, also provide a rich set of user-interface components, written in Java, to assist in the construction of an interactive web-based database application. Since those tools are vendor specific, however, I won't use them for this column and will instead concentrate on the JDBC aspects.

## A Query Tool

For a demonstration applet, I'll create a query tool that allows you to enter any arbitrary SQL query and displays the resulting table in a separate browser window.

I could simply sprinkle JDBC calls throughout the code directly, the way you might have embedded SQL wherever you want to make a query. Database vendors seem to think that developers want to use their interface in this way. It is not good programming practice, however. For example, suppose you want to develop most of the application independent of the database, and merely have a test stub, which simulates a database connection. If there are database calls everywhere, it will be impossible to test the application as a whole without access to the database— a poor strategy. It is better to encapsulate database access in a single class. You could then have a test version of this class and a real version. The test version's methods could return simulated data, whereas the real version's methods return live data.

I've encapsulated my use of java.sql into a class called *QuerySvcsImpl* that implements an interface called *QuerySvcs*. For testing, I am free to have a test version of this class, called *QuerySvcsTestImpl*.

*QuerySvcs* (see Example 1) specifies these methods. Each of these may throw *java.sql.SQLException*, which would indicate an error was detected by the driver. These methods are implemented in *QuerySvcsImpl*. The *initConnection()* method checks if a driver has been instantiated yet (static component), and if one has not, calls *new JetDriver()*— XDB's driver object. This is the last time you have to refer to the client-side driver object. *initConnection()* also performs Example 2.

The *java.sql.DriverManager* class has a static method called *getConnection()* (Example 2) that takes three parameters: a dataset URL, database user id, and password. The URL is the concatenation of the driver protocol, host, and database dataset name. JDBC defines a driver-protocol syntax composed of two parts separated by a colon. The first part is simply "jdbc," while the second part specifies the name of the driver and is vendor specific. In this case, the full protocol is "jdbc:jet.jettp:".

The host is normally the Internet DNS name or the IP address of the database host machine. For example, it could be *www.somewhere.com*, or *205.123.456.789.* If you are testing your database on your own machine, however, you will probably want to specify "loopback" if your machine's TCP/IP stack has loopback support (Windows 95 does not; NT does). However, note that Netscape Navigator requires you to have a real connection, and will not like loopback. If you are testing on Windows 95 and only have a dial-up connection, you can put code into your applet to get your dynamic IP address with the *java.net.InetAddress.getLocalHost()* method and use this IP address in your *getConnection()* call—all this to make the browser happy!

The *getMetaData()* method obtains a database meta data object that can be used to query information about the status of the database. For example, from a *DatabaseMetaData* object you can obtain the dataset URL, the database product name and version, and the name and version of the driver.

The *select()* method executes a SQL query against the database; see Example 3. As you can see, you first have to create a *java.sql.Statement* object before you can perform a query. You then call *executeQuery()* against the *Statement* object, passing it the SQL string to evaluate. The result that is returned is a *java.sql.ResultSet,* which is essentially a currency pointer on the query result. No data is actually returned to the client until the *ResultSet* is traversed.

*vselect()* does all this, then traverses the *ResultSet,* and converts it into a *java.util.Vector* object, with each element of the vector consisting of an array of objects corresponding to the fields in each row of the result. This method is useful if you want to get the whole result and then do something with it, like graph it.

Example 4 is the core of *vselect().* Thus, to find out how many columns are contained in the result, you can get the *ResultSetMetaData* object and then call *getColumnCount().* Note that many of the *ResultSetMetaData* methods can only be called once during the life of any *ResultSet* instance.

What you've discovered is that the *ResultSet.getObject()* method involves a lot of overhead and results in very slow performance. Alternatively, you could use the JDBC type-specific methods of the form *get<type>();*. The disadvantage is that when using these, you must anticipate the type of each column in the result.

The JDBC standard also provides for parameterized precompiled statements (*java.sql.PreparedStatement*), which will result in faster performance if a *select* is executed within a loop.

To put all this together into a query tool, I wrote an applet, Client.java (see Listing One, listings begin on page 113) that instantiates a text field for capturing the input query from the user. A button *handleEvent()* method retrieves this text, and passes it to my *QuerySvcs.select()* method, which returns a *ResultSet.* I then call *toHTML()* with this result set, to generate HTML output for the results window. I use the "javascript:" content protocol to display the generated string of HTML statements: *getAppletContext().showDocument(querySvcs.toHTML(resultSet), "OUTPUT_WINDOW");*. Figure 1 is the completed applet and Figure 2 is a results table generated by it.

## Notes about Data Types

Table 1 lists some of the more useful datatypes. In addition, the Microsoft SQL types DATETIME and SMALLDATETIME are mapped by most ODBC drivers to the ODBC type TIMESTAMP, which maps to *java.sql.Timestamp.* While you cannot create more than one TIMESTAMP field in a record, there is no such restriction on the DATETIME types, and so you can, from your JDBC program's point of view, have more than one *java.sql.Timestamp* object in a record. This is useful for storing *java.util.Date* objects, since *java.sql.Timestamp* derives from *java.util.Date,* and because some databases do not support DATE and TIME, but instead have a DATETIME type. Simply use the *java.sql.Timestamp* type in your program instead of *java.util.Date,* and you will be okay—the *Timestamp* object should work wherever a *Date* object is required. The only consideration is that *java.sql.Timestamp* overrides *java.util.Date*'s date parsing and *toString()* methods, and so it has different (more TIMESTAMP-like) rules for how it parses and displays date and time strings. Note that *java.util.Date* objects can only represent dates as far back as 1970; *java.sql.Timestamp* objects have this same limitation. If you want to represent dates beyond this, you may have to add an offset. Beware of a bug in the parsing of *java.util.Date* objects, which has been fixed in Netscape's version of Java, but not in Internet Explorer. This bug causes a date expressed as 3/1/95 to be interpreted as April 1, 1995 (no joke).

The *Bignum* datatype is very inefficient, and should be used only where monetary values are represented or where exact precision is required.

## The Browser Factor

Sun made a mistake in making the *sql* package part of package *java.* Package *java.sql* will be a standard component of

```
public java.sql.Connection initConnection();
public void closeConnection();
public String toHTML(ResultSet r);
public Vector vselect(String sqline);
public ResultSet select(String sqline);
```

***Example 1:*** *Methods specified by* QuerySvcs.

```
con = DriverManager.getConnection(protocol + "//"
     + host + "/" + dsn, uid, pwd);
dbmd = con.getMetaData();
con.setAutoCommit(true);
```

***Example 2:*** *Executing* initConnection().

```
Statement stmt = con.createStatement();
r = stmt.executeQuery(sqline);
```

***Example 3:*** *Executing the* select() *method.*

```
ResultSetMetaData rmd = r.getMetaData();
int cc = rmd.getColumnCount();

int i = 0;
while (r.next())
{
  i++;
  applet.showStatus("Retrieving row " + i);
  Object[] oa = new Object[cc];
  for (int j = 0; j < cc; j++)
  {
    Object o = r.getObject(j+1);
    if (o == null) o = "";
    oa[j] = o;
  }
  v.addElement(oa);
}
```

***Example 4:*** *The core of* vselect().

package *java* when JDK 1.1 is released. In the meantime, however, the fact that this package is part of package *java* makes it impossible to load it into Netscape Navigator or Internet Explorer in a normal manner. A browser generally will not let you download any component of package *java* from a remote source, for security reasons. Since package *java.sql* is not incorporated into either of the aforementioned browsers (and probably will not be until after JDK 1.1 is released), this means that package *java.sql* is effectively inaccessible to a browser.

A fairly ugly but usable workaround, until 1.1 is released, is to add *java.sql* to the browser's own class library. In the case of Netscape Navigator, as of this writing, these classes are hidden in a ZIP file called java_30. To put *java.sql* into this ZIP archive, you must rename the archive to java_30.zip, open it with a tool such as WinZip, add *java.sql* immediately after the other java packages in the file, and then rename the file back to its original name. For Internet Explorer, the procedure is similar, except that the ZIP file is called classes.zip, and it is in the \Windows\java\classes directory.

Now that you have added *java.sql* to your java_30, users of your applet will have to replace their Netscape java_30 file with yours. You can put it on the web site that serves the applet, with instructions about where to put it after it is downloaded. While this is a little bit cumbersome, it is workable for an intranet application.

An even better alternative is to use a product such as Marimba's Castinet (http://www.marimba.com/), which automatically installs applets on a client system. Castinet does not have the same restrictions that browsers have, because the assumption is that the user has approved the applet and trusts its source. This kind of setup allows you to download new components of package java (such as *java.sql*).

The complete code for the query tool is available electronically (http://www.digitalfocus.com/ddj/code/; or see "Availability," page 3).

## Conclusion

The JDBC interface, combined with a suitable driver, makes it easy to create portable database applications with Java. As of this writing, some installation is required for users to use JDBC applets within a browser, but this obstacle will likely disappear with the next major round of browser releases. There are also alternatives to browsers.

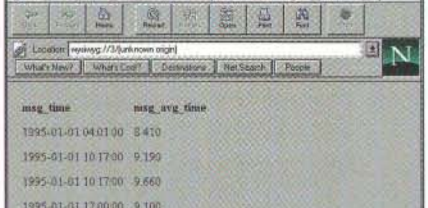
JDBC makes it possible to make enterprise data available, from dynamic and interesting Java applications, without requiring the developer to use precompilers or to write stored procedures: The full power and flexibility of Java is available. Further, these applications can be used with any database for which there are JDBC/ODBC or pure JDBC drivers, and most ODBC driver vendors are developing or now provide JDBC drivers. Java has become a formidable business tool.

**DDJ**

**(Listings begin on page 113.)**

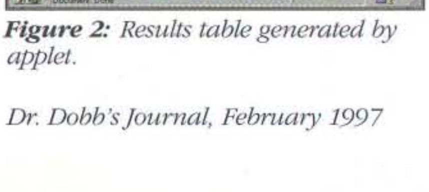
| SQL type | Maps to Java Type | JDBC Method |
|---|---|---|
| VARCHAR | *java.lang.String* | *getString()* |
| CHAR | java.lang.String | *getString()* |
| REAL | *float* | *getFloat()* |
| FLOAT | *double* | *getDouble()* |
| INTEGER | *int* | *getInt()* |
| DATE | *java.sql.Date* | *getDate()* |
| TIME | *java.sql.Time* | *getTime()* |
| NUMERIC, DECIMAL | *java.lang.Bignum* | *getBignum()* |
| TIMESTAMP | *java.sql.Timestamp* | *getTimestamp()* |

***Table 1:*** *Data types.*


Netscape - [file:///D|/column/feb97/Client.html]
File Edit View Go Bookmarks Options Directory Window Help
Location: file:///D|/column/feb97/Client.html
What's New? What's Cool? Destinations Net Search People Software
Execute
Applet Client running

```
select * from MYDATABASE.dbo.MYTABLE;
```

***Figure 1:*** *Completed applet.*


Netscape - [wysiwyg://3/[unknown origin]]
File Edit View Go Bookmarks Options Directory Window Help
Location: wysiwyg://3/[unknown origin]
What's New? What's Cool? Destinations Net Search People
Document: Done

```
msg_time             msg_avg_time
1995-01-01 04:01:00  8.410
1995-01-01 10:17:00  9.190
1995-01-01 10:17:00  9.660
1995-01-01 17:00:00  9.100
```

***Figure 2:*** *Results table generated by applet.*

**WARNING:** Programmers in general have determined that without Opus Make your company's health may be seriously jeopardized.

In today's fast-paced world, time to market is everything. So why risk your company's health using a slow and unreliable build utility? You need Opus Make. It is the fastest, most full-featured, most reliable make utility on the market. Opus Make is designed to efficiently speed up your build process. You will develop your software faster, get it to market sooner and, in return, improve your company's health – and yours. Find out why Andrew Binstock says "*Opus Make is the most compelling reason for not using the make utilities bundled with today's compilers.*"

**Opus Make version 6.1 features include:**

**Makefile Compatibility:** Processes Intersolv Configuration Builder™, Microsoft™ NMake and VC++, Borland Make™, and other makefiles.

**Version Control Support:** Access source files in Microsoft SourceSafe™, Burton Systems TLIB™, Intersolv PVCS™, and MKS Source Integrity™ version control systems.

**Microsoft Visual C++ Support:** Opus Make integrates with VC++ 2.x and 4.x.

**Multiple Platform Support:** DOS, OS/2, WinNT, Win95, AIX, HP/UX, SCO, SunOS, Solaris, Irix, DEC AlphaNT, and others.

**Other features:** Multiple-directory support • Pattern-based inference rules • Multiple targets created from single source • Automatic and in-line response files • Queued shell lines • Object library maintenance • Conditional, looping and include directives • And much more!

Free Technical Support!

**Opus MKMF V6.1 included :** MKMF is our makefile and dependency generator. It quickly builds and maintains your makefiles. It understands C-preprocessor directives and resource compiler files. If you hate building makefiles by hand this is the tool for you!

Opus Software, Inc. • 1032 Irving Street, Suite 439 • San Francisco, CA 94122
Phone: (415) 485-9703 or 800-248-6787 • Fax: (415) 485-9704
Email: info@opussoftware.com • Web: http://www.opussoftware.com

NEW VERSION NOW AVAILABLE

# Opus Make

VERSION 6.1

CIRCLE NO. 323 ON READER SERVICE CARD

# Scheduling Algorithms and NP-Complete Problems

Oleg Kiselyov

The term "NP-complete" refers to a family of problems that are all roughly equivalent; if you could find a fast algorithm to solve one of these problems, the others would quickly follow. NP-complete problems are both extremely common and frustratingly difficult. Many problems of the form "find the best/cheapest/shortest way to do X" are NP-complete, and a fast solution to NP-complete problems would revolutionize such disparate fields as compiler optimization and airline scheduling. However, in lieu of such a breakthrough, the best way to deal with an NP-complete problem is usually to find a trick that will give you a "good enough" approximate solution. This month, Oleg discusses an NP-complete problem that he encountered in trying to schedule a chess tournament, and the approximate solution that made it feasible.

—Tim Kientzle

Many years ago, when I was still a chemist, I was approached by a scientist of the "old school" who didn't like computers. To my surprise, he asked me for help in using computers to schedule a traditional New Year "blitz" chess tournament. That year, quite a few participants were expected, prompting a switch to a Swiss system. In this method of conducting a tournament, the total number of games to be played is specified in advance and is smaller than $n\bullet(n-1)/2$, where $n$ is the number of participants. Consequently, not everybody would end up playing with everybody. To be fair, the scheduler would have to match partners of equal skill, provided they hadn't played already. Keeping track of who played whom and (at the same time) trying to find partners of similar strength would quickly become a pain. That's why he thought a computer might be of some help.

In the actual setting, there were about 20 participants, and 10 rounds were played. In each round, partners played two five-minute games (one person plays white, another black, then they switch). So, one can score 0 (lost both games), 1/2 (lost one, tied the other), and so on up to 2 points (won both). After a round was over, the results were entered into a computer that had to come up with the pairings for the next round. The players scheduled in the next round must not have played each other in previous rounds; moreover, their scores should be as close as possible.

*Oleg is a computer scientist/software developer with Computer Science Corp. (CSC) in Monterey, California. http://pobox.com/~oleg/ftp/.*

For example, consider the tournament in Table 1: Eight people who have already played three rounds. In the first round of the tournament, there are no constraints: No one has played a game, and everyone has the same score (zero). Consequently, any schedule will do; for example, player $2i-1$ plays against player $2i$, with the numbers being assigned randomly. After the round is played, a few leaders emerge. They're pitted against each other in the next round, and so on. After the third round, the leaders are #1 and #5, but they've already played each other. Aiming to match scores as closely as possible, you could pair players #1–#4, #5–#7, and #3–#8. This leaves #6 and #2, but they have already played! You can see the kind of conflicts you have to resolve. With 20 players, the situation becomes very tangled very quickly.

## Handling the Problem

You can think about the scheduling problem in terms of graphs. Consider a graph where each vertex represents a potential pair. For example, vertex 1–2 represents a pairing of players #1 and #2. Initially there are $n\bullet(n-1)/2$ vertices where $n$ is the number of players. As the tournament progresses, you remove vertices that have already played. Two vertices are connected if the two pairs can both appear in the same round; for example, vertices 1–2 and 1–3 are not connected, since player #1 can't play player #2 and player #3 at the same time. Vertices 1–2 and 3–4 are connected. There is a cost associated with each vertex: a disparity of scores of the two players. This cost is updated after each round. Figure 1, for example, shows the graph corresponding to the situation after the third round in our sample tournament. The scheduling problem is reduced to finding a clique of a minimal cost in the graph. That is, find a set of $n/2$ vertices that are all connected with each other (a clique of order $n/2$) such that the total cost (of all vertices in the clique) is minimal. It is obvious that a clique of order $n/2$ means that all $n$ participants will play in the round.

Finding a clique in a graph is, by itself, an NP-complete problem. But it pales in comparison with the requirement of finding a clique of minimal cost. A naive solution would be to consider all possible subsets of $n/2$ vertices of the graph, check to see if they all are connected with each other, compute the total cost, and pick up a subset with the minimal cost. The complexity of this obvious solution is far more than exponential in $n$. For example, in a tournament with 20 participants, you have to choose 10 pairs from 190 possible pairs. There are over $10^{16}$ possible combinations to consider, a formidable number for even the most powerful supercomputer.

Even if you want the exact optimal solution, however, you can do better than this. Assume for a moment that player #1 has stepped out, and we have to schedule a round without him. Obviously, some other player has to take the round off, too. Suppose that for each $i$=2..$n$, you have come up with optimal schedules involving all players but $i$ and 1. In graph terms, you found optimal cliques of order $(n/2)-1$ that don't contain vertices mentioning 1 and $i$. The cost of that clique would be $cost(all-\{1,i\})$. Suppose player #1 has shown up and wants to play the round after all. Rather than rebuilding the schedule from scratch, you can use the

You can waste your time with rookies, theorists, or wannabes. Or you can learn from expert developers who work on real projects every day.

NT FILE SYSTEMS and DEVICE DRIVERS CLASSES

*Learn from the experts the "big guys" call.*

**CLASSES:**

1. DEVELOPING FILE SYSTEMS for WINDOWS NT – 5 day course
2. WINDOWS NT KERNEL MODE DEVICE DRIVERS – 3 day course

| START DATES: | 1 | 2 |
|---|---|---|
| Seattle | 2/10/97 | 2/10/97 |
| Washington, D.C. | - - - | 3/17/97 |
| Palo Alto | 3/31/97 | 4/2/97 |
| Boston | 5/5/97 | 5/5/97 |

**1-888-677-4264**

Call for details and fees.

Or visit OSR's NT World at www.osr.com.

OSR · Open Systems Resources, Inc.

105 Rte. 101A, Suite 19
Amherst, NH 03031 USA
voice: +1 (603) 595-6500
fax: +1(603) 595-6503
e-mail: info@osr.com
web: www.osr.com

EXCEED YOUR EXPECTATIONS

CIRCLE NO. 339 ON READER SERVICE CARD

already-found partial optimal schedules, add players #1 and *i* to them, and select the one with the smallest total cost. For each player *i*, the total cost is *cost(all)* = *cost(all–{1,i})* + *cost({1,i})*, where *cost({1,i})* is the disparity in scores between players *i* and #1. If players *i* and #1 have already played, put *cost({1,i})* to infinity.

This technique is easy to justify: If the optimal schedule matches player #1 against player *i*, then the schedule without these players also has to be optimal (with respect to the remaining players). Otherwise, you can lower the cost of the original schedule, which would contradict the premise that it is optimal. To find the best partial schedules, you can repeat this algorithm recursively. This dynamical programming algorithm—similar to the one used to solve the traveling-salesman problem (TSP)—reduces the complexity somewhat, to the order of $(n-1)2^{n-1}$. The origin of this estimate is easy to see: You need to consider all even subsets of the set of all players. Determining an optimal schedule for a subset (given optimal schedules for smaller subsets computed previously) takes no more than $(n-1)$ operations: adding a cost of a new pair and selecting the smallest term. For $n$=20 players, it should take no more than $10^7$ operations to schedule a round, immensely faster than the aforementioned naive algorithm.

However, this algorithm is fast because it caches the partial schedules. This cache is quite large, several megabytes for $n$=20. At the time, I only had a lowly PDP-11 to work with, so I wasn't able to use even this improved algorithm.

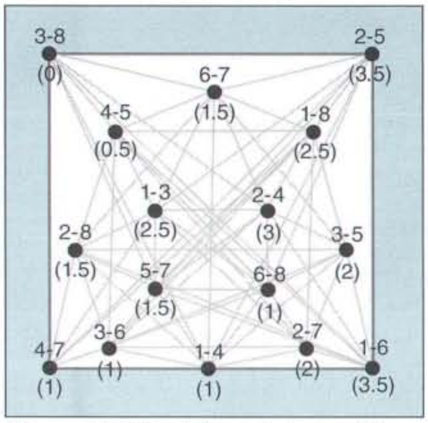


***Figure 1:*** *Scheduling choice graph corresponding to Table 1. Vertex cost is in parentheses.*

## Fast Approximation

In practice, you do not need the perfect schedule—a reasonably good approximation suffices. To speed things up, I used a simpler approach. I first sorted all players by their scores, then tried to pick partners that were neighbors in this sorted list. I started with the player at the top of the list, and walked down the list, picking the first available partner, providing they hadn't played already. Then I'd pick the player with the next-highest score (if still available) and find him a partner, and so on. Since the list is sorted, picking close players from this list gives pairs with similar scores.

```
/* schedule a pair 'pair_no', and then the others */
int schedule_pair(const int pair_no, const int i_start)
{
  if( pair_no >= N_players/2 ) return TRUE;
  for(i=i_start; i<=N_players; i++)
    for(j=i+1; j<=N_players; j++) {
      int ti = Sorted_indices[i], tj = Sorted_indices[j];
      if( allo[ti] == 0 && allo[tj] == 0 && Has_played[ti][tj] == 0 ) {
        /* Try pairing ti and tj */
        allo[ti] = tj; allo[tj] = ti;
        if(schedule_pair(pair_no+1,i+1))
          return TRUE; /* It worked! */
        else
          allo[ti] = allo[tj] = 0; /* It didn't work. */
      }
    }
  return FALSE; /* No valid pairing was found */
}
```

***Example 1:*** *Call* schedule_pair(0,1) *to schedule the entire tournament.*

| | Round 1 | | Round 2 | | Round 3 | | Total |
|---|---|---|---|---|---|---|---|
| Player | Played | Score | Played | Score | Played | Score | Score |
| 1 | 2 | 2 | 5 | 1 | 7 | 2 | 5 |
| 2 | 1 | 0 | 6 | 1 | 3 | 0 | 1 |
| 3 | 4 | 0.5 | 7 | 0 | 2 | 2 | 2.5 |
| 4 | 3 | 1.5 | 8 | 0.5 | 6 | 2 | 4 |
| 5 | 6 | 1.5 | 1 | 1 | 8 | 2 | 4.5 |
| 6 | 5 | 0.5 | 2 | 1 | 4 | 0 | 1.5 |
| 7 | 8 | 1 | 3 | 2 | 1 | 0 | 3 |
| 8 | 7 | 1 | 4 | 1.5 | 5 | 0 | 2.5 |

***Table 1:*** *Three rounds of a sample tournament.*

This greedy strategy of scheduling by adding pairs of players with the closest score may not necessarily lead to the lowest overall cost. For example, after picking all but one pair, you have no choice for the final pair. It may happen that their scores are so different that the total cost is enormous. Worse, this last pair may have already played: Then the entire schedule crumbles. If this is the case, you can back off and try changing the arrangement for the next-to-last pair, hoping that the last pair would be suitable this time. Although building the complete schedule this way may not be optimal, it is close.

Note that the list of players is sorted in descending order, and I start pairing from the top. Most of the backtracking and fiddling is then done with the players near the bottom of the list. That is, even if the final solution may not be optimal as a whole, you err by mismatching losers rather than leaders. The possible inaccuracy pays off handsomely in terms of complexity: It takes only $n \bullet (n-1)/2$ operations to go through the entire list and arrange pairs. Backtracking adds to the complexity, and this could be significant; however, in reality, the convergence is fast. The memory requirements are bounded and small.

Once the schedule is complete, I evaluate its total cost (called *measure* in the code, available electronically; see "Availability," page 3). Then I backtrack once again, trying to find another complete schedule. I repeat this process up to five times, and pick the schedule with the smallest cost. Although the algorithm is based on a greedy strategy, it still tries to find a global optimum. The whole idea is similar to that of approximate TSP algorithms, even though at the time I only had a very vague understanding of TSP.

## Implementation

This fast algorithm was used to schedule the tournament in real time. That is, after a round was finished, the results were entered into the computer. It had to come up with a schedule for the new round while the participants were taking a break. The cumulative tournament results (history) were kept in a matrix *Has_played* and a vector *Scores,* with obvious meaning. The code had provisions to save/restore this history in a file, a precaution against the event of a power surge during the tournament (it happened a few times). Note that the players are numbered 1 through $n$ (so index 0 has a special meaning). A schedule is represented by a vector *allo*: *allo[i],* if not zero, tells who player $i$ is to play with; zero *allo[i]* means that player $i$ is available for allocation. The first step of scheduling is sorting the list of players according to their scores in array *Sorted_indices,* so that *Scores[Sorted_indices[i]] >= Scores[Sorted_indices[j]]* for all $i>j$. The rest of the algorithm can be written as in Example 1.

Function *try_allocate()* (also available electronically) follows this idea almost literally, with the exception that it is not recursive. The first version of the code was written in DEC Basic on the PDP-11, which doesn't support recursion. Even when I later rewrote the program in C, I stuck with the faster iterative approach.

The only variables that define an arrangement of a pair are the indices $i$ and $j$ of the players in the sorted list. You can keep these indices for each arranged pair in special arrays, *si* and *sj,* accessed through an index *level.* This makes it easy to back off the arrangement to the previous level(s). The scheduling process is made of a series of actions: Pick up a suitable $i$, pick up $j$, arrange the next pair (that is, advance the "level"), or back off and try the next $j$; after that, try the next $i$. The action to execute depends on the previous action and some tests (availability of players). This is exactly how a finite automaton runs. The scheduler function in the listing implements this automaton literally.

The C compiler I initially used was pre-ANSI C. I have recently beautified the code, mainly adding *void.* The code runs, and even the user interface for a VT100 terminal works. The program had been used for several years and, I think, helped convince my professor that computers aren't all evil.

**DDJ**

*The longest continuously advertised C product in the history of... mankind.*

PC-lint

*for C/C++*
Version 7.0
presents Bug # 514

```
#include <stdio.h>
#define PEPPERONI       1
#define MUSHROOMS       2
#define ONIONS          4
#define PEPPERS         8
#define ANCHOVIES    0x10
#define SAUSAGE      0x20

double pizza( unsigned order )
    {
    double cost = 5.00;
    if( order & (SAUSAGE | PEPPERONI) == 0 )
        printf( "meatless\n" );
    while( order )
        {
        if( order & 1 ) cost += 1.00;
        order >>= 1;
        }
    return cost;
    }
```

*This program to process pizza orders attempts to record whenever a meatless pizza is ordered. But it doesn't seem to be working. Why not? Call if you need a hint, or refer to our web page at* `http://www.gimpel.com`.

**PC-lint for C/C++** will catch this and many other bugs. It will analyze a mixed suite of C and C++ modules to uncover bugs, glitches, quirks and inconsistencies.

**Version 7** of PC-lint breaks new ground with inter-statement value tracking for both automatic variables and class data members. Taking clues from assignment statements, initializers and conditional expressions it can detect out-of-bound subscripts and potential null pointer uses. As an enabling technology, almost 100 standard functions are rigorously checked. Also macros are subject to increased scrutiny, checking for unparenthesized parameters, unparenthesized bodies and repeated arguments having side-effects.

**Plus Our Traditional C/C++ Warnings:** Uninitialized variables, inherited non-virtual destructors, strong type mismatches, inadvertent name-hiding, suspicious expressions, etc., etc.

**Full C++ Support -** PC-lint for C/C++ is based on the ARM and is tracking the latest ANSI/ISO draft including exceptions and templates. It supports both Borland and Microsoft C/C++.

*PC-lint for C/C++ $239*

Numerous compilers/ libraries supported. Runs on MS-DOS (Optional built-in 386 DOS extender), OS/2, NT and Windows 95. This price is subject to increase.

*FlexeLint for C/C++*

The same great product for other operating systems. Runs on all Unix systems, VMS, mainframes, etc. Distributed in shrouded C source form. Call for pricing.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426

**CALL TODAY (610) 584-4261 Or FAX (610) 584-4266**

30 Day Money-back Guarantee.

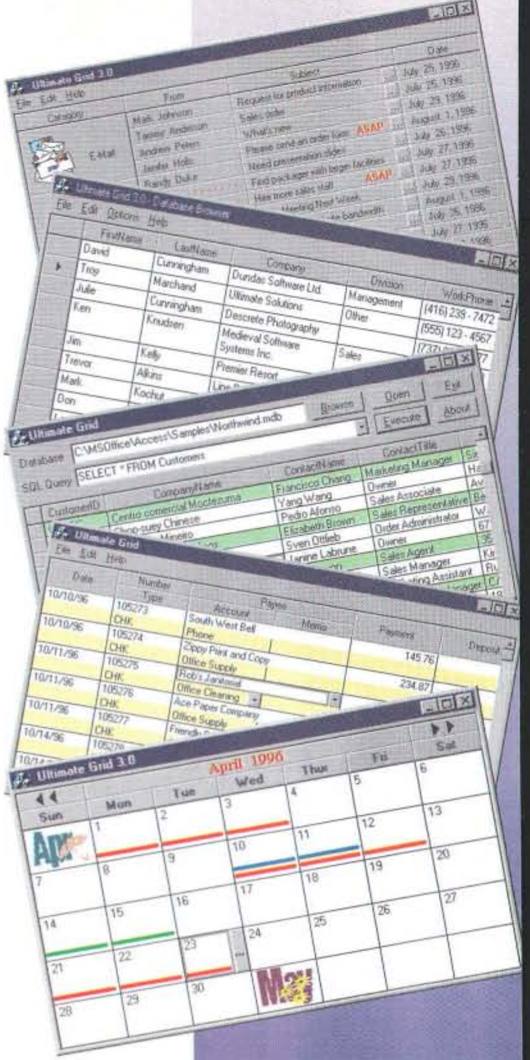
PA add 6% sales tax.

PC-lint and FlexeLint are trademarks of Gimpel Software

# Ultimate Grid™

## 100% MFC Object Oriented Design

The Premier Grid control for MFC/C++ development, Ultimate Grid provides an outstanding 100% MFC object oriented design. Unparalleled power, Unparalleled features, Ultimate performance.



## Solid. Powerful. Professional.

Ultimate Grid was designed from the ground up as a solid, professional tool to satisfy your development needs. Based on powerful MFC classes, Ultimate Grid can be easily used as a CWnd, CView, CDialog, dialog resource, or a pop-up!

Ultimate Grid includes all the features you expect, and an extensive list of powerful new features you can depend on. It connects easily to standard data sources such as DAO, ODBC, SQL, CodeBase, and c-tree plus. And, you can very easily connect to proprietary or other 3rd party databases, or use the built-in memory manager for pre-loading.

The professional edition includes full source code and allows you to develop custom data sources and new cell types, gives you enhanced debugging capability and greater development flexibility.

Ultimate Grid compiles right into your app; no ActiveX control or DLL to include! It's 100% Royalty-free, just like all Dundas products, and includes a 30 day money-back guarantee.

Call today, or visit our WWW site and find out for yourself why developers the world over are flocking to Ultimate Grid and making it the cornerstone of their development projects.

| | |
|---|---|
| Entry-Level | $149.00 |
| Professional (w/ full source code) | $349.00 |

Generic C++ and ActiveX versions are also available. Call or visit our Web site for full details.

# Ultimate TCP/IP™

## Internet/Intranet Client and Server Development Kit

| | | |
|---|---|---|
| ✓ POP3 | ✓ FTP | ✓ Clients and Servers |
| ✓ SMTP | ✓ DNS | ✓ Custom Protocols |
| ✓ HTTP | ✓ And More | ✓ NT Services |

Ultimate TCP/IP allows you to design full-featured, powerful FTP, HTTP, DNS, Finger, SMTP/POP3 applications, as well as proprietary protocols quickly and easily! Modify our sample server and client code to get exactly what you want instantly! Easily create multi-threaded Internet server applications that run as true Windows NT services.

Whether you are developing for the intranet or internet, Ultimate TCP/IP will be your trusted guide. Our well-constructed C++ classes allow you to solve both sides of the client/server equation.

| | |
|---|---|
| Entry-Level | $199.00 |
| Professional (w / full source code) | $499.00 |

## Dundas Software

**1 800 463 1492**

- 30 day money-back satisfaction guarantee.
- Visa, MasterCard, American Express, check, money order.
- We will deliver via Internet, Compuserve, mail or courier.
- All prices in US dollars

| | |
|---|---|
| Sales: | 416 239 7472 |
| Fax: | 416 239 2183 |
| EMail: | sales@dundas.com |
| WWW: | www.dundas.com |

Dundas Software Ltd.
- 500 - 4800 Dundas Street West, Etobicoke, Ontario, Canada, M9A 1B1
- 670 - 240 Portage Rd., Lewiston, New York, USA, 14092

www.dundas.com

Windows is a registered trademark of Microsoft Corp. All other products are trademarks or registered trademarks of their respective owners.

CIRCLE NO. 279 ON READER SERVICE CARD

# Applying *CSplitterWnd* Internals

George Shepherd and Scot Wingo

In our December 1996 column, we took a look at the internal structures of the popular *CSplitterWnd* class. This month, we'll show how you can apply knowledge of the *CSplitterWnd* internals to solve a *CSplitterWnd* problem that MFC programmers frequently encounter.

When using static or dynamic *CSplitterWnds* in applications, you may want to change the splitter position programmatically. *CSplitterWnd* provides one undocumented solution that is easy to use—simply add a menu/toolbar item that has ID_WINDOW_SPLIT as the identifier. When a user selects this menu/toolbar item, it will place the splitter into a mode where the user has to change the position of the splitter bar. The mouse cursor moves to the splitter bar, and the splitter bar is also put into tracking mode: The user can choose its position by moving the mouse and clicking to lock in the new position. Let's see how this seemingly magical feature works.

ID_WINDOW_SPLIT is a global MFC resource ID defined in AFXRES.H. The resource ID is used in VIEWCORE.CPP; see Example 1. *OnUpdateSplitCmd()* disables the menu/toolbar item if the parent of the view is not a splitter window. The real work is done by *CView::OnSplitCmd()*. Listing One (listings begin on page 114) is the pseudocode for this member function.

*OnSplitCmd()* first verifies that the parent of the view handling the command is a *CSplitterWnd* by calling *GetParentSplitter()*. If the view is inside a *CSplitterWnd*, it then ASSERTs that the splitter bar is not already in tracking mode, which should be impossible since the menu/toolbar item should be disabled by *OnUpdateSplitCmd()*. MFC has many such ASSERTs that check the validity of states that should be True. This catches the problem of a user mistakenly changing the behavior of *OnUpdateSplitCmd()* to allow the splitter bar to enter tracking mode twice. Once everything is validated, *OnSplitCmd()* calls *CSplitterWnd::DoKeyboardSplit()*. Listing Two is the pseudocode for this member function.

First, *DoKeyboardSplit()* does some if/else checking to find out which part of the *CSplitterWnd* needs to be "activated." For example, if there are two splits open, *DoKeyboardSplit()* will need to move both bars, so it sets the value of *ht* to *splitterIntersection1*. Once the value of *ht* is determined, *DoKeyboardSplit()* calls *StartTracking()*—the same function that is called when the user presses the mouse button to "grab" a splitter bar. Finally, after calling *StartTracking()*, *DoKeyboardSplit()* makes some calculations and then programmatically moves the mouse cursor over the splitter bar (or intersection) being manipulated by calling *SetCursorPos()*.

While the ID_WINDOW_SPLIT trick is helpful and easy to use, there are common *CSplitterWnd* situations it does not address.

## Not So Static *CSplitterWnd*

Many of today's applications have a Microsoft Explorer look-and-feel with a vertical static splitter that separates a tree view on the left, and list view on the right. The problem with using *CSplitterWnd* for this functionality is that in static mode you cannot change the orientation of the splitter bar. For example, what if you want to provide users with the ability to switch from a vertical presentation to a horizontal presentation?

In our previous column, we showed that *CSplitterWnd* has key data members that maintain the row/column information for the class; see Table 1. We'll need to manipulate these members to implement the static-bar flipping functionality. Since these *CSplitterWnd* data members are all protected, the first step is to create a *CSplitterWnd* derivative so that you can freely access the data members and encapsulate your *CSplitterWnd* enhancement

```
ON_COMMAND_EX(ID_WINDOW_SPLIT, OnSplitCmd)
ON_UPDATE_COMMAND_UI(ID_WINDOW_SPLIT,
      OnUpdateSplitCmd)
```

***Example 1:** Resource ID is used in VIEWCORE.CPP.*

| Data Member | Description |
|---|---|
| m_nRows/m_n | Number of rows and columns currently being displayed in the *CSplitterWnd.* |
| m_nMaxRows/ m_nMaxCols | Maximum number of rows/columns as specified in the call to *Create()* or *CreateStatic()* |
| m_pColInfo | Array of *CRowInfo* with one element for each column in the *CSplitterWnd.* This value is fixed in static splitter windows and varies in dynamic splitter windows |
| m_pRowInfo | Array of *CRowColInfo* with one element for each row in the *CSplitterWnd.* This value is fixed in static splitter windows and varies in dynamic splitter windows. |

***Table 1:** CSplitterWnd data members.*

*Scot is a cofounder of Stingray Software. He can be contacted at ScotWi@aol.com. George is a senior computer scientist with DevelopMentor and can be contacted at 70023.1000@compuserve.com. They are the coauthors of* MFC Internals *(Addison-Wesley, 1996).*

in a stand-alone class. The enhanced *CSplitterWnd* is called "*CDobbsSplitter*," and its declaration is presented in Listing Three. From Listing Three you can see that a couple of data members have been added to *CDobbsSplitter* to help with the implementation of the *FlipSplit()* function. Table 2 lists these data members.

Listing Four is the implementation of the *CDobbsSplitter* class. The main functionality provided by the class comes from the *FlipSplit()* member function. This member function determines whether the *splitter-window* is currently split horizontally or vertically, then calls either *SplitVertically()* or *SplitHorizontally()* to "flip" the split. The logic for detecting whether the current split is horizontal or vertical is based on the number of rows. For example, if the number of rows is greater than the number of columns, then the split is horizontal. If the number of columns is greater than the number of rows, then there is a vertical split.

The *SplitHorizontally()/SplitVertically()* functions actually perform the manipulations of the split. They do this by programmatically swapping the numbers of rows (*m_nRows/m_nMaxRows*) and columns (*m_nCols/m_nMaxCols*). After swapping the values of these data members, the row and column information arrays are also swapped (*m_pColInfo* and *m_pRowInfo*). Next, the ID given to each pane is swapped so that there are no focus problems using the *SetDlgCtrlID()* API. Finally, whenever you programmatically change a splitter window, you must call *RecalcLayout()* to cause the splitter window to update.

Notice also that *SplitHorizontally()* and *SplitVertically()* make a call to the *UpdatePanes()* member function. This member updates the *m_pColInfo* and *m_pRowInfo* size information arrays with new data based on the new split. This data is calculated using the previous split ratios and sizes of the panes.

## Conclusion

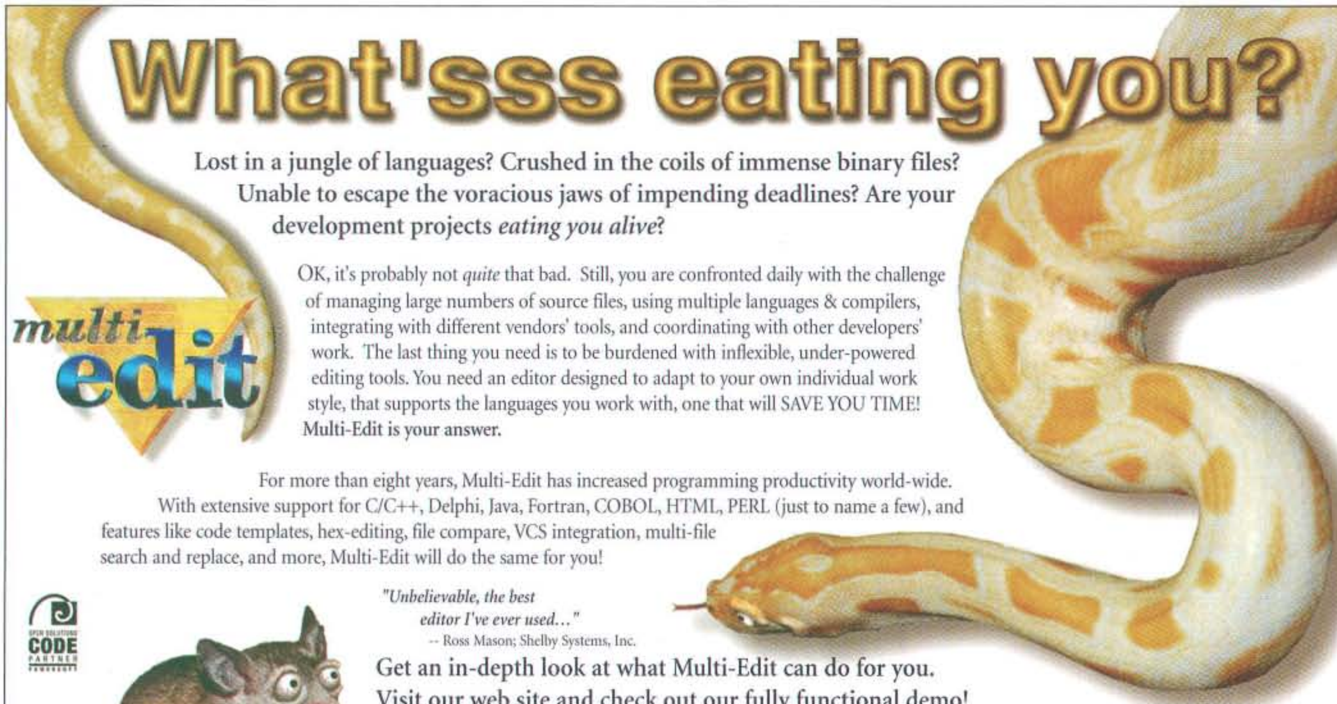
By applying some of our undocumented *CSplitterWnd* knowledge, you have added new functionality to the *CSplitterWnd* and given end users the flexibility to change the appearance of a program that uses a static splitter. You can take this example and extend it to swap panes, or work in a *CDialog*. For a real challenge, you might even try to make a dynamic *CSplitterWnd* that can have more than two rows and columns. A sample application that further illustrates how this additional functionality can be implemented is available electronically (see "Availability," page 3).

**DDJ**

**(Listings begin on page 114.)**

| Data Member | Description |
|---|---|
| m_bPanesSwapped | Boolean that indicates if you are in the original state (False) or in a "flipped" state (True) so that you can change back and forth easily. |
| m_nSplitRatio | When flipping the split from one position to another, be sure that you keep the ratio of the two panes exactly the same. This data member is used to store the ratio. |
| m_nSplitResolution | Since you don't need to be 100 percent exact about the ratio of the panes, use this multiplier (500 works well) to convert the ratio into a large integer to avoid floating-point arithmetic. |

***Table 2:*** CDobbsSplitter *data members.*



What'sss eating you?

**Lost in a jungle of languages? Crushed in the coils of immense binary files? Unable to escape the voracious jaws of impending deadlines? Are your development projects *eating you alive*?**

multi edit

OK, it's probably not *quite* that bad. Still, you are confronted daily with the challenge of managing large numbers of source files, using multiple languages & compilers, integrating with different vendors' tools, and coordinating with other developers' work. The last thing you need is to be burdened with inflexible, under-powered editing tools. You need an editor designed to adapt to your own individual work style, that supports the languages you work with, one that will SAVE YOU TIME! **Multi-Edit is your answer.**

For more than eight years, Multi-Edit has increased programming productivity world-wide. With extensive support for C/C++, Delphi, Java, Fortran, COBOL, HTML, PERL (just to name a few), and features like code templates, hex-editing, file compare, VCS integration, multi-file search and replace, and more, Multi-Edit will do the same for you!

*"Unbelievable, the best editor I've ever used..."*
-- Ross Mason; Shelby Systems, Inc.

**Get an in-depth look at what Multi-Edit can do for you. Visit our web site and check out our fully functional demo!**

**www.amcyber.com**
**1-800-899-0100**

**Suggested Retail Price $199**

1830 West University Dr. Ste.112
Tempe, AZ, 85281
Voice: (602) 968-1945
Fax: (602) 966-1654

AMERICAN CYBERNETICS

*Multi-Edit is a trademark of American Cybernetics, Inc. Other products are trademarks of their respective publishers. Delphi is a trademark of Borland. Developer tested only. American Cybernetics stands behind all of its products with a 30 day Money Back Guarantee. No animals were eaten by other animals during the making of this ad.*

**CIRCLE NO. 143 ON READER SERVICE CARD**

## Listing One

```
// ----- midiin.h
#ifndef MIDIIN_H
#define MIDIIN_H

#include <mmsystem.h>

class MidiIn    {
    void* m_pFunc;
    HMIDIIN hMidiIn;
    short int numDevices;
    short int currDevice;
public:
    MidiIn();
    ~MidiIn();
    void RegisterMIDIFunction(void* pFunc)
        { m_pFunc = pFunc; }
    void DeviceList(CListBox* dlist);
    BOOL ChangeDevice(short int device);
};
#endif
```

## Listing Two

```
// ---- midiin.cpp

#include "stdafx.h"
#include "midiin.h"

MidiIn::MidiIn()
{
    m_pFunc = 0;
    hMidiIn = 0;
    currDevice = -1;
    numDevices = midiInGetNumDevs();
}
MidiIn::~MidiIn()
{
    if (hMidiIn != 0)    {
        midiInStop(hMidiIn);
        midiInClose(hMidiIn);
    }
}
void MidiIn::DeviceList(CListBox* dlist)
{
    ASSERT(dlist != 0);
    MIDIINCAPS icaps;
    for (short int i = 0; i < numDevices; i++)  {
        midiInGetDevCaps(i, &icaps, sizeof(icaps));
        dlist->AddString(icaps.szPname);
    }
}
BOOL MidiIn::ChangeDevice(short int device)
{
    ASSERT(device < numDevices);
    ASSERT(m_pFunc != 0);
    if (device != currDevice)    {
       if (hMidiIn)     {
            midiInStop(hMidiIn);
            midiInClose(hMidiIn);
            hMidiIn = 0;
        }
        HMIDIIN hIn;
        if (midiInOpen(&hIn,device,(DWORD) m_pFunc,0,CALLBACK_FUNCTION) != 0) {
            currDevice = -1;
            return FALSE;
        }
        hMidiIn = hIn;
        midiInStart(hMidiIn);
        currDevice = device;
    }
    return TRUE;
}
```

## Listing Three

```
// ----- midiout.h
#ifndef MIDIOUT_H
#define MIDIOUT_H

#include <mmsystem.h>

class MidiOut    {
    HMIDIOUT hMidiOut;
    short int numDevices;
    short int currDevice;
public:
    MidiOut();
    ~MidiOut();
    void DeviceList(CListBox* dlist);
    void SendEvent(DWORD dwEvent);
    void StartMessage();
    void TimingMessage();
    void StopMessage();
    BOOL ChangeDevice(short int device);
    void CloseDevice();
};
inline void MidiOut::SendEvent(DWORD dwEvent)
{
    if (hMidiOut != 0)
        midiOutShortMsg(hMidiOut, dwEvent);
}
#endif
```

## Listing Four

```
// ---- midiout.cpp

#include "stdafx.h"
#include "midiout.h"

MidiOut::MidiOut()
{
    hMidiOut = 0;
    // ---- test for MIDI output devices
    numDevices = midiOutGetNumDevs();
    currDevice = -1;
}
MidiOut::~MidiOut()
{
    CloseDevice();
}
void MidiOut::CloseDevice()
{
    if (hMidiOut != 0)  {
        // --- close the device
        midiOutClose(hMidiOut);
        hMidiOut = 0;
    }
}
BOOL MidiOut::ChangeDevice(short int device)
{
    ASSERT(device < numDevices + 1);
    if (device != currDevice)    {
        CloseDevice();
        // --- open the new device
        HMIDIOUT hOut;
        if (midiOutOpen(&hOut, device, 0, 0L, 0L) != 0) {
            currDevice = -1;
            return FALSE;
        }
        currDevice = device;
        hMidiOut = hOut;

    }
    return TRUE;
}
void MidiOut::StartMessage()
{
    if (hMidiOut != 0)  {
        DWORD mmsg  = 0xfa;
        midiOutShortMsg(hMidiOut, mmsg);
    }
}
void MidiOut::TimingMessage()
{
    if (hMidiOut != 0)  {
        DWORD mmsg  = 0xf8;
        midiOutShortMsg(hMidiOut, mmsg);
    }
}
void MidiOut::StopMessage()
{
    if (hMidiOut != 0)  {
        DWORD mmsg  = 0xfc;
        midiOutShortMsg(hMidiOut, mmsg);
    }
}
void MidiOut::DeviceList(CListBox* dlist)
{
    ASSERT(dlist != 0);
    MIDIOUTCAPS ocaps;
    for (short int i = 0; i < numDevices; i++)  {
        midiOutGetDevCaps(i, &ocaps, sizeof(ocaps));
        dlist->AddString(ocaps.szPname);
    }
}
```

## JAVA Q&A

## Listing One

```
/* Copyright (c) Digital Focus, 1996, 1997.
 * This code may be used for non-commercial purposes.
 * Digital Focus gives no warrantee or guarantee.
 */

import java.applet.*;
import java.awt.*;
import java.io.*;
import java.util.*;
import java.net.*;
```

*(continued on page 114)*

*(continued from page 113)*

```
/** Query Tool.  */
public class Client extends Applet
{
    TextArea queryWindow = new TextArea(3, 40);
    Button go = new Button("Execute");
    QuerySvcsImpl querySvcs;

    /** Initialize the applet. */
    public void init()
    {
   add(queryWindow);
  add(go);
   // Instantiate query services
   try
   {
  querySvcs = new QuerySvcsImpl(getParameter("protocol"),
getParameter("host"), getParameter("dsn"), getParameter("uid"),
getParameter("pwd"));
   }
   catch (java.sql.SQLException ex)
   {
  ex.printStackTrace();
  showStatus(ex.getMessage());
  return;
   }
    }
    /** Handle the button press event. */
    public boolean handleEvent(Event e)
    {
   if ((e.target == go) && (e.id == Event.ACTION_EVENT))
   {
  java.sql.ResultSet r = null;
  String html = null;
  URL url = null;

  try
  {
 r = querySvcs.select(queryWindow.getText());
 html = querySvcs.toHTML(r);
 System.out.println(html);
 url = new URL("javascript:'" + html + "'");
 getAppletContext().showDocument(url, "results_window");
  }
  catch (Exception ex)
  {
 ex.printStackTrace();
 showStatus(ex.getMessage());
 return true;
  }
   return true;
   }
   else return super.handleEvent(e);
    }
}
```

## UNDOCUMENTED CORNER

### Listing One

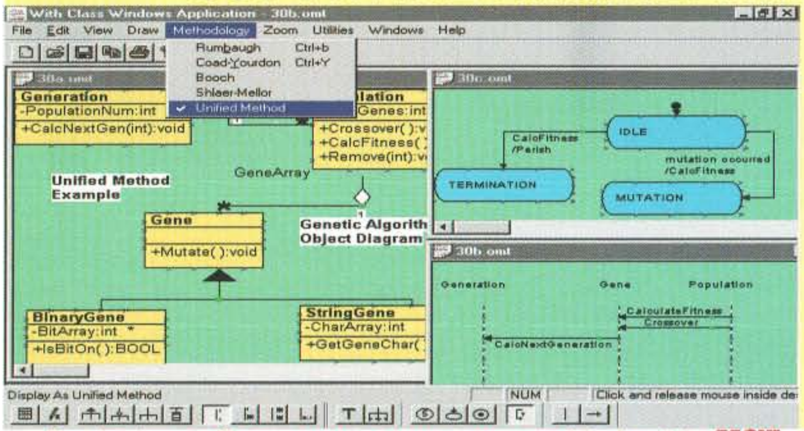
```
BOOL CView::OnSplitCmd(UINT)
{
    CSplitterWnd* pSplitter = GetParentSplitter(this, FALSE);
    if (pSplitter == NULL)
        return FALSE;
    ASSERT(!pSplitter->IsTracking());
    pSplitter->DoKeyboardSplit();
    return TRUE;
}
```

### Listing Two

```
BOOL CSplitterWnd::DoKeyboardSplit()
{
    int ht;
    if (m_nRows > 1 && m_nCols > 1)
        ht = splitterIntersection1; // split existing row+col
    else if (m_nRows > 1)
        ht = vSplitterBar1;         // split existing row
    else if (m_nCols > 1)
        ht = hSplitterBar1;         // split existing col
    else if (m_nMaxRows > 1 && m_nMaxCols > 1)
        ht = bothSplitterBox;       // we can split both
    else if (m_nMaxRows > 1)
        ht = vSplitterBox;          // we can split rows
    else if (m_nMaxCols > 1)
        ht = hSplitterBox;          // we can split columns
    else
        return FALSE;               // can't split

   // start tracking
    StartTracking(ht);

    CRect rect;
```

# THE POWER OF TRUE O-O TECHNOLOGY

## *SOFTWARE ENGINEERING FOR WINDOWS*



- Reverse Engineer Java, C++ & Delphi code to generate graphical designs
- Supports Booch, Rumbaugh, Shlaer-Mellor, Coad-Yourdon & the Unified Modeling Language
- Generate Code from your models for any language
- Reverse Relational Database into entity relationship diagram
- Unique scripting language for custom code generation & reporting
- **INCREDIBLY AFFORDABLE!**

*Designing software was never supposed to be this **EASY!***

**WITH CLASS** offers a complete Analysis, Design and Modeling solution for all phases of your project. Reverse engineer your code to pinpoint design errors before you continue coding. Built-in scripts will help you get started *quickly* - **WITH CLASS** will have you designing in under 30 minutes! Other Features include:

• OLE support • Error checking • ASCII Import/Export • Custom Color and **Fonts**

Call us at 908.668.4779 for your **FREE** Evaluation copy today!

MODELING OOA/D MICROGOLD DEVELOPMENT TOOLS DESIGNED FOR YOU.

**Register to win a FREE copy of WITH CLASS from our web page: www.microgold.com**

***ONE TOOL. ONE PRICE. ALL YOUR NEEDS.***

**MICROGOLD SOFTWARE, INC.**
**Tel: 908.668.4779 Fax: 908.668.4386**
**http://www.microgold.com**
**email: 71543.1172@compuserve.com**

CIRCLE NO. 332 ON READER SERVICE CARD

```
        rect.left = m_rectTracker.Width() / 2;
        rect.top = m_rectTracker.Height() / 2;
        if (m_ptTrackOffset.y != Ø)
            rect.top = m_rectTracker.top;
        if (m_ptTrackOffset.x != Ø)
            rect.left = m_bTracking2 ? m_rectTracker2.left :m_rectTracker.left;
        rect.OffsetRect(-m_ptTrackOffset.x, -m_ptTrackOffset.y);
        ClientToScreen(&rect);
        SetCursorPos(rect.left, rect.top);

        return TRUE;
}
```

## Listing Three

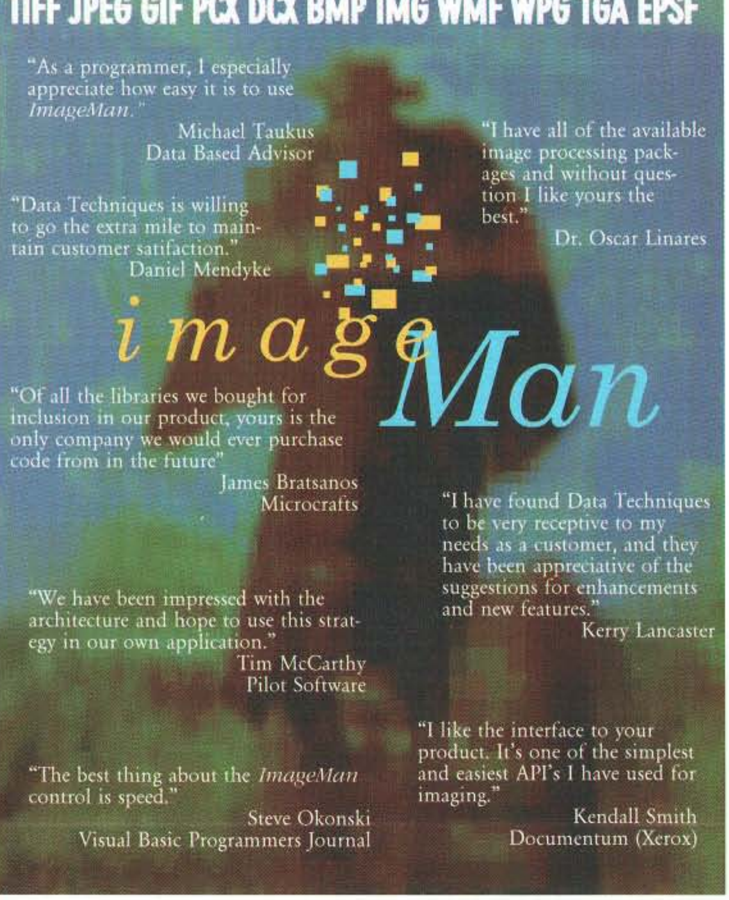
```
class CDobbsSplitter : public CSplitterWnd
{
// Construction
public:
    CDobbsSplitter();

// Attributes
public:
    BOOL m_bPanesSwapped;
    int m_nSplitRatio;
    int m_nSplitResolution;
// Operations
public:
  void SetSplitRatio( int nRatio );
  BOOL IsSplitHorizontally() const;
  BOOL IsSplitVertically() const { return !IsSplitHorizontally(); }
  BOOL ArePanesSwapped() const { return m_bPanesSwapped; }
protected:
    BOOL UpdateSplitRatio();
    BOOL UpdatePanes( int cx, int cy );
    BOOL UpdatePanes();
public:
void FlipSplit();     //DDJ
    void SplitVertically();
    void SplitHorizontally();
    // Implementation
public:
    virtual ~CDobbsSplitter();
protected: // Generated message map functions
    afx_msg void OnSize(UINT nType, int cx, int cy );
    DECLARE_MESSAGE_MAP()
};
```

## Listing Four

```
CDobbsSplitter::CDobbsSplitter()
{
    //Intialize extended state.

   m_nSplitRatio = -1;
    m_bPanesSwapped = FALSE;
    nSplitResolution = 1;
}
BEGIN_MESSAGE_MAP(CDobbsSplitter, CSplitterWnd)
    ON_WM_SIZE()
END_MESSAGE_MAP()
void CDobbsSplitter::SetSplitRatio( int nRatio )
{
    m_nSplitRatio = nRatio;
}
BOOL CDobbsSplitter::IsSplitHorizontally() const
{
    ASSERT(( m_nRows > 1 ) != ( m_nCols > 1 ));
    ASSERT( max( m_nRows, m_nCols ) == 2 );
    return ( m_nCols > 1 );
}
void CDobbsSplitter::SplitHorizontally()
{
    if( IsSplitHorizontally())
        return;
    ASSERT( m_nCols = 1 );
    ASSERT( m_nRows = 2 );
    CWnd* pPane = GetDlgItem( IdFromRowCol( 1, Ø ));
    ASSERT( pPane );
    // swap the H/V information
    m_nMaxCols = m_nCols = 2;
    m_nMaxRows = m_nRows = 1;
    CRowColInfo* pTmp = m_pColInfo;
    m_pColInfo = m_pRowInfo;
    m_pRowInfo = pTmp;
   // change the last pane's ID reference
    pPane->SetDlgCtrlID( IdFromRowCol( Ø, 1 ));
    ASSERT( GetPane( Ø, 1 )->GetSafeHwnd() == pPane->GetSafeHwnd() );
    if( UpdatePanes())
        RecalcLayout();
}
void CDobbsSplitter::SplitVertically()
{
    if( IsSplitVertically())
        return;
    ASSERT( m_nCols = 2 );
    ASSERT( m_nRows = 1 );
    CWnd* pPane = GetDlgItem(IdFromRowCol( Ø, 1 ));
```

*(continued on page 116)*

**TIFF JPEG GIF PCX DCX BMP IMG WMF WPG TGA EPSF**



"As a programmer, I especially appreciate how easy it is to use *ImageMan*."
Michael Taukus
Data Based Advisor

"I have all of the available image processing packages and without question I like yours the best."
Dr. Oscar Linares

"Data Techniques is willing to go the extra mile to maintain customer satifaction."
Daniel Mendyke

"Of all the libraries we bought for inclusion in our product, yours is the only company we would ever purchase code from in the future"
James Bratsanos
Microcrafts

"I have found Data Techniques to be very receptive to my needs as a customer, and they have been appreciative of the suggestions for enhancements and new features."
Kerry Lancaster

"We have been impressed with the architecture and hope to use this strategy in our own application."
Tim McCarthy
Pilot Software

"I like the interface to your product. It's one of the simplest and easiest API's I have used for imaging."
Kendall Smith
Documentum (Xerox)

"The best thing about the *ImageMan* control is speed."
Steve Okonski
Visual Basic Programmers Journal

## *ImageMan®—Trusted By More Of The Companies You Trust*

*ImageMan* is used in more off-the-shelf applications than any other imaging library available, including applications like:

| | |
|---|---|
| *LotusApproach* | *My Advanced Label Designer* |
| *CorelDraw* | *My Advanced Mailing List* |
| *DataEase Express* | *Avery LabelPro* |
| *Zylab ZyIndex* | *Gupta SQLWindows* |
| *Milestones, Etc. (Kidasa)* | *More...* |

These companies didn't choose *ImageMan* because of slick advertisements or meaningless guarantees. They chose *ImageMan* because it meets their real-world image processing needs. They chose *ImageMan* because it's the fastest, easiest to use, and most comprehensive solution they could find. And they chose *ImageMan* because *ImageMan* is clearly a technically superior product. Take the time to compare *ImageMan's* demo with those of our competitors, compare the APIs, and you'll come to the same conclusion that some of the best programmers in the industry have already reached: *ImageMan* is the only choice for creating truly world-class products. That's why *ImageMan* is a *Visual Basic Programmer's Journal* **Readers' Choice** award recipient, and that's why *ImageMan* users are so devoted.

*ImageMan* has always been offered with an unconditional 90-day money-back guarantee (yes, even the 32-bit versions). We can offer this extraordinary guarantee because of our extremely high level of confidence in our product. Can our competitors say the same?

*ImageMan* is available royalty-free in DLL or VBX/OCX format for Windows 3.1, Win32s, and Windows NT. *Source code available.*

800-955-8015

Visual Basic Programmer's Journal 1994 Award Readers' Choice

340 Bowditch St, #6
Burnsville, NC 28714
(704) 682-4111
Fax: (704) 682-0025
BBS: (704) 682-4356

D·T·I
DATA TECHNIQUES, INC.

**CIRCLE NO. 80 ON READER SERVICE CARD**

(continued from page 115)

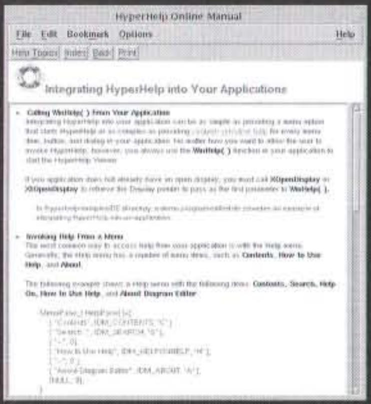
```
    ASSERT( pPane );
    // swap the H/V information
    m_nMaxCols = m_nCols = 1;
    m_nMaxRows = m_nRows = 2;
    CRowColInfo* pTmp = m_pColInfo;
    m_pColInfo = m_pRowInfo;
    m_pRowInfo = pTmp;
    // change last pane's ID reference (no need to change ID for first one)
    pPane->SetDlgCtrlID( IdFromRowCol( 1, Ø ));
    ASSERT( GetPane( 1, Ø )->GetSafeHwnd() == pPane->GetSafeHwnd() );


    if( UpdatePanes())
        RecalcLayout();
}
void CDobbsSplitter::FlipSplit()
{
    if( IsSplitHorizontally())
        SplitVertically();
    else
    {
        ASSERT( IsSplitVertically());
        SplitHorizontally();
    }
}
int CDobbsSplitter::UpdateSplitRatio()
{
    CRowColInfo* pPanes;
    int czSplitter;
    if( IsSplitHorizontally())
    {
        pPanes     = m_pColInfo;
        czSplitter = m_cxSplitter;
    }
    else
    {
        pPanes     = m_pRowInfo;
        czSplitter = m_cySplitter;
    }
    TRACE( "CDobbsSplitter::UpdateSplitRatio: 1:/i, 2:/i\n",
           pPanes[Ø].nCurSize, pPanes[1].nCurSize);
    if(( pPanes[Ø].nCurSize != -1 ) &&
       ( pPanes[Ø].nCurSize + pPanes[1].nCurSize != Ø ))
    {
        m_nSplitRatio = nSplitResolution * pPanes[Ø].nCurSize /
                    ( pPanes[Ø].nCurSize + pPanes[1].nCurSize + czSplitter );
    }
    TRACE("m_nSplitRatio=/i\n", m_nSplitRatio );
    return m_nSplitRatio;
}
BOOL CDobbsSplitter::UpdatePanes()
{
    CRect rcClient;
    GetClientRect( rcClient );
    return ( !rcClient.IsRectEmpty() &&
             UpdatePanes( rcClient.Width(), rcClient.Height()));
}
BOOL CDobbsSplitter::UpdatePanes( int cx, int cy )
{
    TRACE("UpdatePanes: cx=/i, cy=/i\n", cx, cy );
    CRowColInfo* pPanes;
    int cz;
    if( IsSplitHorizontally())
    {
        pPanes = m_pColInfo;
        cz     = cx;
    }

    else
    {
        pPanes = m_pRowInfo;
        cz     = cy;
    }
    BOOL bRes = UpdateSplitRatio();
    if( m_nSplitRatio >= Ø )
    {
        ASSERT( (ULONG)m_nSplitRatio * cz /
            nSplitResolution <= (ULONG)INT_MAX );
       pPanes[Ø].nIdealSize =
            int( (ULONG)m_nSplitRatio * cz / nSplitResolution );
    }
    return bRes;
}
void CDobbsSplitter::OnSize( UINT nType, int cx, int cy )
{
    TRACE("CDobbsSplitter::OnSize: cx:/i, cy:/i\n", cx, cy );
    if(( nType != SIZE_MINIMIZED )&&( cx > Ø )&&( cy > Ø ))
        UpdatePanes( cx, cy );
    CSplitterWnd::OnSize( nType, cx, cy );
}
```

**DDJ**

ONLINE HELP

For UNIX & OpenVMS

HyperHelp™ is the UNIX and OpenVMS equivalent of the industry-standard Microsoft Windows WinHelp. Use your favorite Windows help authoring tool to create online hyperlinks, pop-up definitions, graphics, full-text searching, powerful macros, and a wealth of other features. You can even utilize the same RTF, HPJ and graphic files used to create your Windows help.



Integrating HyperHelp into Your Applications

- Calling WinHelp( ) From Your Application
  [illegible]
- Invoking Help From a Menu
  [illegible]

HyperHelp also supports FrameMaker MIF, HTML, SGML and ASCII text as authoring file formats for online help. Best of all, the HyperHelp Viewer can be distributed royalty-free to your customers. For more information, call (203) 438-6969, fax (203) 438-5013 or download a free demo from *ftp.bristol.com*. In Europe call: +31 (0)33 450 50 50 or fax +31 (0)33 450 50 51.

Now Supports HTML!

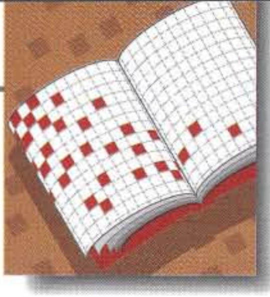
All trademarks are the property of their respective holders.

Web: www.bristol.com
Email: info@bristol.com

BRISTOL
T·E·C·H·N·O·L·O·G·Y

CIRCLE NO. 170 ON READER SERVICE CARD

# Into the Future



Phil Mitchell

The prolific French novelist Balzac was once asked how he was able to write so many books. He replied, "I never use labor-saving devices." It's unclear what 19th-century gadgets Balzac was avoiding, but apparently the sentiment is timeless: The biggest "labor-saving" device that should never have been used, according to Thomas Landauer, is about $3 trillion worth of computers applied in business over the past two decades. This appalling conclusion sets the stage for his engaging and lucid book, *The Trouble with Computers.*

**The Trouble with Computers:**
**Usefulness,**
**Usability, and**
**Productivity**

*Thomas K. Landauer*
MIT Press, 1996
440 pp., $15.00
ISBN 0-262-62108-8

**The Future**
**of Software**

*Edited by Derek Leebaert*
MIT Press, 1996
320 pp., $13.50
ISBN 0-262-62109-6

### The Trouble with Computers

Fiscally speaking, the trouble with computers is that economists can't seem to show that they help the bottom line. Of course, there were the early, easy gains from number crunching; and there are certain specialized applications, notably CAD, where computers have made the unthinkable possible. But in terms of the service industries that increasingly dominate our economy, where information technology (IT) was supposed to trigger vast efficiencies for ordinary workers, there's no sign of a productivity gain. To the contrary, productivity growth in the era of minicomputer and desktop applications has notably and unsettlingly slowed. Landauer goes so far as to argue that investment in IT was a major *cause* of this downturn, because, in many cases, it was wasted money. But while causality is hard to prove (and Landauer is no economist), what's hard to argue with is that the huge productivity gains forecast for computerization, gains on par with mechanization, simply haven't materialized.

Landauer's sedulous examination of the "productivity paradox" prompts the question: What went wrong? He's glad you asked. For despite the plodding econometrics of the early chapters, Landauer is a man with a thrilling story to tell. When he examines the reasons for IT's failure—reasons that include the hidden costs of training and maintenance, the absence of standards and interoperability, and mismanagement and misapplication of technology—he finds the overriding, fundamental flaw to be the failure of designers to create useful and usable software. But the thrilling part is that he's sure this is a problem that can be fixed, and he's convinced that if it *were* fixed, the ensuing productivity gains would truly revolutionize society.

This is not, however, a book of wishful thinking. Landauer headed up one of the human factors/user-interface research groups at Bell Labs in the '80s. He brings to the topic of usability the rigor of empirical psychology and the extensive experience of an insider from one of the few industries with a major IT success story to tell. It is a powerful combination. His critique of usability, across a broad spectrum of applications, is relentless and insightful. The quirkiest example is his application of the concept of random reinforcement schedules to computer users. (It is well known in academic psychology that a pigeon—or a person—who is rewarded for some behavior according to a regular and predictable scheme will behave logically: If the rewards are frequent enough, the behavior is maintained; if they become too infrequent, the behavior stops. But if the reward schedule becomes random and infrequent, the result is obsessive behavior that does not extinguish. See online Help.)

Landauer's real mission, though, is to show us how to do better—how to create that useful and usable software to produce the gains we've been looking for. In the final third of the book, he discusses a number of examples where usability design was done right. In particular, his detailed discussion of an electronic book project at Bellcore is fascinating. From these examples, and from numerous empirical studies, Landauer draws some remarkable conclusions.

For instance, studies repeatedly show that computer users exhibit wide variability in efficiency. Among expert manual typists, the worst performer might be about 30 percent slower than the best; among expert word processor users, the difference jumps to 400 percent! (This effect is notoriously present among programmers.)

There are a number of ways that computers seem to magnify variability; but the startling conclusion that Landauer presents is that it is possible to design systems that eliminate much of this variability. Whereas a poor interface tends to separate the good users from the bad, a well-designed interface can bring most people up to a high level of performance. Contrary to current practice, Landauer proposes that systems should be designed not to maximize users' choice and flexibility, (users' intuitions being as bad as programmers'), but to offer them the clear-cut best way.

There's nothing mysterious about design for usability. It centers around (surprise!) testing applications with real users in an iterative design process. But there are some useful facts to know. For example, it doesn't take a large and expensive study to do rigorous usability testing. The average interface has around 40 defects in need of repair; having two naïve users evaluate it will find about half the flaws, and six evaluations will typically find about 90 percent. Additionally, software that doesn't go through careful

*Phil, president of NeoCortek, can be contacted at phil_mitchell@neocortek.com.*

usability testing is pretty much guaranteed to lower user productivity. Designers who hope to do otherwise would do well to read this book.

### *The Future of Software*

Landauer's book left me wondering what the big software manufacturers would have to say about these matters. Conveniently, in *The Future of Software,* Derek Leebaert invited representatives of IBM, Microsoft, Lotus, Novell, Intel, and DEC, as well as assorted industry insiders, to give *their* views on the usefulness of software, present and future.

The official agenda of the book is the "software problem," the fact that our remarkable progress in hardware seems to outstrip our ability to design remarkable (read: intelligent) software. But beneath this problem statement is the productivity paradox, as several contributors make explicit. Microsoft's director of enterprise computing feels our vast investment in personal computers has not delivered value to individuals and organizations. (But fear not, it will soon.)

*The trouble with computers is that economists can't seem to show that they help the bottom line*

Asking these companies to describe the future of software is a bit like asking the National Cheese Council to describe the future of hors d'oeuvres. The results are pretty *un*surprising, with convergent views on the ascendance of networking, open standards, and business reengineering on the basis of enhanced information flow. Still, it's intriguing to listen to what the big guys want you to think they're thinking.

For instance, Microsoft lumps mainframes and PCs together as relics of hierarchical, assembly-line organizational thinking. It's the client/server model, supporting distributed databases and seamlessly integrated application suites, that will enable businesses to restructure around processes: There will be no more isolated order-entry clerks— a salesperson will shepherd the entire process from order to fulfillment.

Lotus goes a step further, suggesting that the current state of computing is positively medieval, and that workgroup computing will usher in a humanistic renaissance. The writers take some hard shots at the current state of usability, averring that "...most of the problems with computers [are] that computer people [talk] too much to their computers and not to other people." Rather than productivity enhancements for the individual worker, "intelligent communications" will empower workers and revolutionize decision-making processes at all levels: Relationships with customers, suppliers, and business partners will become efficient, mutual, and collaborative.

The contributions from Digital and Novell have a different emphasis: Both envision a future in which the end user (and who better?) is able to create his/her own software. Open standards and reusable components will put an end to the need for expert programmers to create novel applications. The interesting part of DEC's piece describes two current efforts at standardization: a user-driven effort in which the Japanese telecom giant NTT demanded that its suppliers transition from proprietary to open standards, and an industry-driven effort (the SPIRIT project), in which American, European, and Japanese telecom and IT companies are collaborating on such standards. Novell, on the other hand, emphasizes its Visual AppBuilder, in which software components will enable vendors or end users to design complex custom business applications in a single day.

### Conclusion

The problem with these futurist manifestos is that each ends without ever addressing the software problem; these marvelous futures assume a new level of software intelligence without indicating how it will be achieved. Somehow, networked computers are going to lead to natural-language translators, and open standards are going to make object components trivial to use— but no one explains how. The one article that tries to fill this gap is Gustave Essig's essay on natural-language processing and artificial intelligence. Essig, a philosopher and computer telephony entrepreneur, believes that new knowledge representations, based on functional insights from the neural and cognitive sciences, are about to make fully intelligent "naturalware" possible. We'll see. In the meantime, it's hard to take another round of big promises seriously. I'd settle for some decent usability testing.

**DDJ**

Metrowerks' Code Warrior development system has been made available for Windows. The Windows-hosted compiler can generate executable code for Windows 95/NT x86, MacOS 68K, and MacOS PowerPC. It supports C, C++, Object Pascal, and Java. The system is available in several different bundles, ranging from the professional Gold package ($399.00, including three updates over the next year) to packages targeted at beginning programmers ($79.00) and students ($119.00). Developers who purchase the initial Developer's Release will receive automatic updates to the final version when it becomes available.
Metrowerks Inc.
2201 Donley Drive, Suite 310
Austin, TX 78758
800-377-5416
http://www.metrowerks.com/

Infrastructures for Information has announced a desktop edition of its Standard SGML Support System (S4), a real-time SGML language processor. Delivered as middleware, S4 can be embedded in existing software such as word processors, composition tools, databases, and the like. The S4-Desktop API implementation includes over 70 function calls to support SGML-tag searching and navigation, attribute searching and navigation, hypertext navigation, and contact extraction. The S4-Desktop Developer's Package also includes a number of sample applications—such as a set of Word macros that allow the use of Microsoft Word as the composition engine—and sample code that shows how to decompose SGML instances into information components, store them in an SQL database, and reassemble them for export as valid SGML files. Also included is an Adobe Exchange Plug-In that automates the process of redoing SGML hotlinks and TOCs into PDF files.

S4-Desktop is delivered as a set of DLLs in the Windows environment and as a set of shared libraries in the Macintosh and UNIX environments. It costs $1500.00 for the Developers License, plus $149.00 per seat run-time charge. Three versions—16- and 32-bit for PCs and Macintosh—are available.
Infrastructure for Information Inc.
344 Bloor Street West, Suite 400
Toronto, ON
Canada M5S 3A7
416-920-6489
http://www.i4i.org/

Source Dynamics is shipping Source Insight 2.1 for Windows 95/NT, a tool designed for large projects that has been enhanced to edit large, multiple-module Java applications. Source Insight 2.1 also provides enhancements to its C/C++ and assembler program-editing capabilities. Introductory pricing for Source Insight 2.1 is $249.00.
Source Dynamics
22525 SE 64th Place, Suite 260
Issaquah, WA 98028
206-557-3630
http://www.sourcedyn.com/

Thunder & Lightning has released Web Factory Author 3.0, a web-page authoring tool. Its features include dynamic WYSIWYG and HTML simultaneous editing, image import, and conversions. The company also announced three add-on software modules: ImagEditor for image editing; Preview Library for content management; and Dreamcatcher for web-site management. These modules may be added in any combination to the foundation product, Web Factory Author 3.0. The integrated suite of Web Factory software is available in a single product, Web Factory Pro Image 3.0. Thunder & Lightning is distributing its products exclusively through downloads from its site and other sites on the World Wide Web. Web Factory Pro Image is available for Windows 95/NT for $249.00 per copy. You can lease Web Factory Pro Image for $59.95 for three months.
Thunder & Lightning Company
6540 Lusk Boulevard, Suite C-212
San Diego, CA 92121
619-643-5550
http://www.tlco.com/

Catalyst SocketWrench is a WinSock custom control in the form of a VBX, usable with Visual Basic 3.0, the 16-bit edition of Visual Basic 4.0, or any software that supports the 2.0 VBX specification. The SocketWrench VBX works with any TCP/IP app that provides a WinSock DLL compliant with the Version 1.1 specification. With SocketWrench you can create a client or server applications using either TCP or UDP protocols.

SocketWrench also includes a dialer custom control in the form of a VBX that interfaces with Microsoft's Remote Access Services. The SocketWrench RAS control allows you to create a dialer for any host configured with RAS. All in all, SocketTools contains 13 custom controls that simplify Internet and Intranet application development. It includes a browser control as well as controls for other TCP/IP protocols such as FTP, SMTP, POP3, NNTP, DNS, ICMP/PING, RAS, RLIB, Image Viewer, TELNET, ANSI, and SocketWrench. Catalyst SocketTools sells for $347.00 and is royalty free.
Catalyst Development Corp.
56925 Yucca Trail, Suite 254
Yucca Valley, CA 92284
619-228-9653
http://www.catalyst.com/

Chrysalis-ITS has released its Luna data-encryption token and an SDK for developing secure systems for electronic commerce. Luna is a security token that uses its on-board 32-bit RISC processor and up to 1 MB of memory for computing encode and decode data and provide digital signature verification. The SDK consists of two Luna tokens and Luna card readers, an ISA connection into a PC-based development system, PKCS#11 software and documentation, CK source code, and 60 days telephone/e-mail support. The Luna card is a standard Type II PCMCIA or PC card that plugs into a standard PC card interface on laptop computers. Luna PC cards are available in quantity for 30 day delivery. The single-unit price for the Luna card is $350.00. The Luna Developer's Kit is priced at $10,000.00.
Chrysalis-ITS Inc.
380 Hunt Club Road, Suite 200
Ottawa, ON
Canada K1V 1C1
613-731-6788
http://www.chrysalis-its.com/

Zinc Software has introduced Zinc DataConnect (ZDC), a database middleware tool that provides C++ access to relational databases. ZDC provides database independence by supporting different database back ends with a single object-oriented C++ database API. Unlike ODBC, ZDC translates its own database-independent function calls to native API calls for individual databases. ZDC can be hosted on Windows 3.1/95/NT, OS/2, Macintosh, DOS (Real, 16-bit, 32-bit), or UNIX. You can buy a Core Library and Database Access Modules for each database for which support is needed. Price includes full source code, with no royalty or run-time fees. The Core Library costs $199.00 for PC platforms and $399.00 for UNIX. The Core Library with all Database Access Modules costs $2999.00 for PCs and $5999.00 for UNIX.
Zinc Software
405 South 100 East
Pleasant Grove, UT 84062
801-785-8900
http://www.zinc.com/

Passport Corp. announced Passport IntRprise, a tool that lets you build applications that use Java front ends for Internet/intranet deployment, yet support transaction processing, multitier architectures, and fault tolerance for enterprise computing. Passport IntRprise supports Windows 3.1/95/NT, UNIX, and Java. A single developer's license is $8995.00.
Passport Corp.
Mack Centre III
140 East Ridgewood Avenue
Paramus, NJ 07652
201-634-1100
http://www.passport4gl.com/

Eastern Systems has announced TestWeb, a software-testing tool that nonintrusively tests applications. It supports regression testing, benchmarking, and compatibility testing on x86 based PCs, embedded systems, and systems that utilize proprietary user-input devices. TestWeb uses a single, comprehensive C-based toolkit. Eastern Systems also announced TestBed 5 for Windows 95/NT, an integrated toolset that provides programming standards verification, structured programming verification, complexity metric measurement, variable crossreference, unreached code reporting, static control-flow tracing, and procedure call analysis.
Eastern Systems Inc.
P.O. Box 1087
Westboro, MA 01581
508-366-3223
http://www.easternsystems.com/

Interactive Software Engineering (ISE) announced EiffelWeb, a library that provides scripting tools for developing CGI scripts using Eiffel. EiffelWeb is supported on platforms such as Windows 3.x/95/NT, SunOS, Solaris 2.4, HP 9000, IBM RS/6000, DEC Alpha OSF/1, Data General Avion, Silicon Graphics, Linux, UnixWare, and SCO.
Interactive Software Engineering
270 Storke Road, Suite 7
Santa Barbara, CA 93117
805-685-1006
http://www.eiffel.com/

WebManage Technologies has announced the release of its NetIntellect log-analysis tool. NetIntellect is a 32-bit Windows-based application that operates with any Common Log File or Combined Log File from any web, intranet, or Proxy server. NetIntellect lets you save or view reports and graphs as HTML, Microsoft Word, Microsoft Excel, Lotus, text files, and so on, instead of only HTML or Microsoft Word files. You can also access 30 predefined reports, or customize them as needed (tables, 2-D and 3-D graphs) by using various filters for hits, access time, bytes transferred, host name, file name, and domain name. A trial version of NetIntellect is available by accessing the WebManage web site. The retail price for a single-user version of NetIntellect is $149.00.
WebManage Technologies Inc.
70 West Red Oak Lane
White Plains, NY 10604-3602
914-697-7555
http://www.webmanage.com/

MapInfo has introduced SpatialWare, a server technology that provides spatial information from a database source. SpatialWare uses MapInfo mapping technology with Oracle's relational database system to let users store, manage, and analyze complex spatial data. SpatialWare is available for Sun Solaris and SCO UnixWare and supports Oracle v7.1.6 or v7.2.3. Prices range from $25,000 for a work group configuration to $200,000 for an enterprise server implementation.
MapInfo Corp.
One Global View
Troy, NY 12180-8399
518-285-6000
http://www.mapinfo.com/

SCH Technologies has announced a DEC Alpha version of its SystemWatch systems-management tool. SystemWatch manages disk space, memory, swap space, CPU utilization, host availability, database performance/availability, system/application processes, and system logs. SystemWatch's event management lets it take corrective actions without administrator intervention. SystemWatch sells for $695.00 with one year of maintenance. The program runs on DEC Alpha, Sun, HP, and IBM platforms.
SCH Technologies
895 Central Avenue
Cincinnati, OH 45202-1961
513-579-0455
http://www.sch.com/

AutoTester has announced AutoTester Distributed Test Facility (DTF) 2.0 for distributed testing of Windows and OS/2 GUI and client/server applications. You can use AutoTester DTF to execute and monitor tests created for Windows 3.x/95/NT and OS/2 across a network for load stress and performance testing. A combination of tests may be scheduled for simultaneous or synchronized playback across a combination of network machines. Prices begin at $15,000.00.
AutoTester Inc.
8150 N Central Expressway, Suite 1300
Dallas, TX 75206
214-368-1196
http://www.autotester.com/

Blue Sky Software RoboHELP 4 is a Help authoring tool that supports all Windows Help platforms and HTML-based Help standards. RoboHELP 4 turns Microsoft Word 7 into an authoring tool capable of creating professional Windows Help, HTML-based Help, printed documentation, and intranet/Internet web sites, using a single source. RoboHELP 4 is available for $499.00.
Blue Sky Software
7777 Fay Avenue, Suite 201
La Jolla, CA 92037
800-459-2356
http://www.blue-sky.com/

The KL Group has released JClass BWT, a library of components that lets Java developers create GUI applications and applets. JClass BWT includes Outliner (hierarchical tree control), Image Button, Tab Manager, Multicolumn List, Header, Label, and Scrollbar. The JClass BWT components provide advantages over Java's AWT such as consistent look and feel across platforms, dynamically configurable resources, and additional components. JClass BWT costs $49.00, or $199.00 with source.
KL Group Inc.
260 King Street East
Toronto, ON
Canada M5A 1K3
416-594-1026
http://www.klg.com/

ImageFX has announced PhotoPRO, a 32-bit ActiveX toolkit for adding a multitude of image display and processing capabilities to desktop and Internet applications. PhotoPRO provides image manipulation capabilities, TWAIN image acquisition, image printing capabilities, and image retrieval via HTTP. PhotoPRO is royalty free and has a suggested retail price of $499.00.
ImageFX
3021 Brighton-Henrietta Road
Rochester, NY 14623
716-272-8030
http://www.imagefx.com/

IntegrationWare has announced Speed Daemon for Delphi, a Delphi source-code profiler. The utility produces statistics for each function such as the number of times a function is called, the total time spent executing the function relative to other pieces, and average time per call. Speed Daemon works with all Delphi 1.0 and 2.0 applications, including DLLs and OLE automation servers developed in Delphi and single-threaded and multithreaded applications. Speed Daemon does not require any changes to a project or source code. The regular price is $89.00.
IntegrationWare Inc.
Deerfield Tech Center
111 Deer Lake Road, Suite 109
Deerfield, IL 60015
888-773-1133
http://www.integrationware.com/

**DDJ**

## THE ADVERTISER INDEX

| ***RS#*** | ***Advertiser Name*** | ***Page #*** |
|---|---|---|
| 124 | Abraxas Software, Inc. | 26 |
| 17 | Accusoft | 17 |
| 297 | Addison Wesley | 118 |
| 167 | Aladdin Software | 62 |
| * | Al Stevens Cram Course CD-ROM | 90 |
| 143 | American Cybernetics | 112 |
| * | Annasoft | 48 |
| 338 | Applied Microsystems | 25 |
| 180 | Blue Water Systems | 40 |
| 67 | Borland International | C4 |
| 68 | Bristol Technology | 21 |
| 170 | Bristol Technology | 116 |
| * | Centura | 25 |
| 305 | Communication Intelligence | 63 |
| 329 | Compaq Computer Corp | 94–95 |
| 324 | Computer Management Group | 102 |
| 79 | Concurrent Sciences Inc | 56 |
| 131 | Connect Tech Inc | 79 |
| 80 | Data Techniques Inc. | 115 |
| 151 | Datalight | 58 |
| * | *Dr. Dobb's* Alternative Languages CD-ROM | 49 |
| 125 | Distinct Corporation | 61 |
| * | *Dr. Dobb's* File Formats CD-ROM | 109 |
| * | *Dr. Dobb's* Distributed Objects CD-ROM | 96 |
| 84 | FairCom | 4 |
| 285 | Geodesic Systems Inc. | 86 |
| 603 | Geodyssey Limited | 123 |
| * | Gimpel Software | 109 |
| 87 | Greenleaf Software | 84 |
| * | Henry Holt & Company Inc | 106 |
| 600 | InfoMagic | 123 |
| * | InstallShield Corporation | 65 |
| 288 | KL Group Inc | 66 |
| 90 | LEAD Teachnologies | 67 |
| 150 | Leading Design | 86 |
| * | MathTools | 81 |
| 146 | MDBS | 43 |
| 332 | MicroGold Software | 114 |
| * | Microquill | 48a/b |
| * | Microsoft | C2–1 |
| * | Microsoft | 39 |
| * | Microsoft | 73 |
| 94 | MKS | 47 |
| * | NuMega Technologies | 5 |
| * | NuMega Technologies | 102 |
| 168 | O'Reilly & Associates | 45 |
| * | On Time Marketing | 54 |
| 339 | Open Systems Resources | 108 |
| 323 | Opus Software | 105 |
| 162 | Pacific Softworks | 52 |
| 340 | Pacific Softworks | 53 |
| * | Paradigm Systems | 57 |
| 294 | Parasoft Corp | 27 |
| 99 | Phar Lap Software, Inc. | 59 |
| 118 | Popkin Software & Systems Inc. | 23 |
| 289 | Portable Graphics Inc | 79 |
| 295 | Powersoft | 13 |
| 121 | Premia | 41 |
| 102 | Prentice Hall | 64 |
| ** | Product Debut | 123–124 |
| ** | Programmer's Marketplace | 125–127 |
| 128 | Programmer's Paradise | 14–15 |
| * | Pure Atria CD-ROM | 32a/b |
| 134 | QNX | 51 |
| * | Quinn-Curtis | 11 |
| * | Raima Corp | 99 |
| 104 | Rational Software Corporation | 7 |
| 215 | Rogue Wave Software | 28–29 |
| 331 | Routerware | 54 |
| * | SD Conference West | 122 |
| * | SD Web Development | 83 |
| 105 | Sequiter Software Inc | C3 |
| 164 | Silicon Graphics Inc. | 37 |
| 309 | Snowbound Software | 49 |
| 318 | Soffront Software Inc. | 26 |
| 336 | Softbank WebSource | 77 |
| * | Softel vdm | 123 |
| 333 | Software Builders | 93 |
| 286 | Stingray Software | 87 |
| 257 | SunSoft | 31 |
| 195 | Symantec Corp | 8–9 |
| 301 | Tom Sawyer | 85 |
| 605 | Tetradyne Software Inc. | 124 |
| 112 | UnderWare | 12 |
| * | *Unix Review* | 33 |
| 155 | V Communications Inc. | 46 |
| * | Vireo Software | 66 |
| 174 | Visix Software Inc. | 35 |
| 602 | WCSC | 123 |
| 274 | Willows Software | 44 |
| 604 | WIBU Systems | 124 |
| * | World Wide Web Directory | 127 |

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

*The advertiser prefers to be contacted by phone; consult ad.
**See individual ads for RS#.

### *Advertising Sales Offices*

| | | |
|---|---|---|
| **Mid-Alantic/Southeast** | | |
| **John Moon** | Sales Manager | (617) 235-8577 |
| **Nicole Glazen** | Sales Associate | (617) 235-8568 |
| **California/Southwest** | | |
| **Paul Miller** | Sales Manager | (415) 655-4183 |
| **Aimee Einstein** | Sales Associate | (415) 655-4221 |
| **Northwest/Mountain/Texas** | | |
| **Carrie Mills** | Sales Manager | (415) 655-4196 |
| **Poppy Mussallem** | Sales Associate | (415) 655-4301 |
| **Northeast/Northcentral** | | |
| **Brenner Fuller** | Sales Manager | (603) 588-3058 |
| **Software Careers** | | |
| **Lilly Gutierrez** | National Account Manager | (415) 655-4193 |

# You could learn a lot about software development from a ten-year-old.

Register for Software Development '97 West, March 31-April 4, Moscone, San Francisco

That's right. SD is ten years old, and you can still count on us to keep you up to speed on the entire spectrum of development technologies, issues and techniques.

From C++ to Delphi to Java, if you use it to build applications, you'll use it more effectively after five prolific days at Software Development '97.

This year, as always, you'll gain fresh insights directly from the creators of the most advanced technologies. Learn new ways to meet your development challenges, through the industry's most intensive technical conference. Master the fine art of managing the development process. And browse through a vast supermarket of more than 500 high-powered tools.

To receive your SD '97 West catalog, call us at 800.441.8826 or 415.905.2702, send us e-mail at sd97west@mfi.com, or visit our web site. You'll be amazed at how much ingenuity a ten-year-old could contribute to your development team.

SOFTWARE DEVELOPMENT

More tools. More intelligence.

www.sd97.com

DR. DOBB'S JOURNAL

# PRODUCT DEBUT

11950 N. Highway 89
Flagstaff, AZ 86004

**InfoMagic**

**1-800-800-6613**
**Tel +1-520-526-9565**
**Fax +1-520-526-9573**
**orders@infomagic.com**

**BSDisc 2-CD Set** ....................$35
*NetBSD & FreeBSD unix-clone implementations*

**Delphi Tools CD-ROM** ............$35
*Shareware & add-ons for Borland's Delphi product!*

**Hobbes OS/2 4-CD Set** ...........$35
*OS/2 demos, games, utilities, ports, fixes & updates*

**InfoMagic Workgroup Server** .......................................$75
*High-performance file & print services to PC & Mac clients using the Linux OS*

**Internet Tools CD-ROM** ..........$30
*Networking tools & utilities for PC & UNIX*

**JAVA CD-ROM** ........................$25
*JDK, demo applets, HotJAVA browsers and on-line docs*

**LINUX Developers Resource CD-Set** ....................$25
*Red Hat, Slackware, Debian, sources, on-line docs, more!*

**LINUX Toolbox (CD-ROM + manual)** ................$45
*Perfect for beginners! Includes CD-ROM set and manual*

**Moo-Tiff CD-ROM (for LINUX)** ..............................$99
*OSF/MOTIF-compatible GUI for PC's running LINUX!*

**Moo-Tiff CD-ROM (for FreeBSD)** ..........................$99
*OSF/MOTIF-compatible GUI for PC's running FreeBSD!*

**Mother of PERL 2-CD Set** ......$35
*Utility language for UNIX, DOS, NT, MAC and more!*

**Novell Netware Tools CD-ROM** ........................$35
*Tools & utilities for Novell Netware!*

**Source Code CD-ROM** ...........$30
*4.4 BSD-Lite2, MACH, InterViews, JPEG, GNU sources*

**Standards 2-CD Set** ................$30
*Domestic & international networking standards*

**TCL/TK CD-ROM** ....................$35
*General purpose scripting language*

**TeX 2-CD Set** ..........................$35
*Powerful tools for creating professional-quality typesetting*

**Visual Basic CD-ROM** .............$35
*Programs & modules for building VB applications*

**Webmaster Tools CD-ROM — Vol 1** ..................$35
*Tools for designing and publishling web pages*

**WinSite CD-ROM Set**...$35
*Shareware for Windows 3.1, NT and Windows95*

**Windows95.com CD-ROM**...............$35
*The definitive collection of Windows95 software!*

**X-11 Files 2-CD Set**.........$35
*complete source distribution for X11R6 & XFree86*

***Call, Fax or send e-mail MC, VISA and AMEX accepted***

**Visit our website: http://www.infomagic.com**

CIRCLE NO. 600 ON READER SERVICE CARD

Supports Windows 95 & Windows NT

## SERIAL I/O

**The ONLY Serial Communication Libraries, DLLs, & Tools You Will Ever Need For Windows 3.x, Windows 95, Windows NT, and MS-DOS. Accept No Less Than The Best.**

http://www.wcscnet.com/home.htm

**COMM-DRV/LIB™**
**Professional Serial I/O Libraries & DLLs for Windows 3.x, Windows 95, NT, & MSDOS**

- Supports ALL languages, tools, or applications that can call the Windows API.
- **Complete source code to all libraries & DLLs.**
- Same API for Windows 3x, Windows 95, NT, & MS-DOS.
- High level Hayes compatible modem functions.
- **X, Y, Zmodem file transfers on multiple ports.**
- Asynchronous & timed callback to user functions on different serial communication events(modem signals, buffer counts, character reception, and more).
- Supports **Visual** C/C++, ANSI C/C++, QuickBasic, Visual Basic (MSDOS & Windows), Assembly, Access,
- Transmit data from user callback on any event.
- Extensive scanning of input character stream.
- Support most intelligent/all dumb multiport cards(Arnet, AST, Boca, Cyclades, Digiboard, GTEK, & many more).

**COMM-DRV/DOS™**
**MS-DOS Serial I/O TSRs & Utilities**

- True DOS device driver that allows serial ports to be opened like files under MS-DOS & Windows.
- FOSSIL interface for all supported(Supports all BBS, Mailers, etc.[PCBoard, Wildcat, Frontdoor, Doorway, Renegade, etc.])
- Provides Int14h, Int21h, & DAM interfaces.
- Compatible with Desqview and Windows.
- Real time serial port monitoring to screen and disk.
- Mini-BBS for file transfer/Spawnable file transfer engine.
- Supports most intelligent & non-intelligent multiport cards.

**COMM-DRV/VxD™**
**Ultra High Speed Serial I/O VxD For Windows**

- Windows(95 & 3.x) DLLs with API calls into the 32bit VxD.
- MS-DOS high speed libraries for calling VxDs from a DOS box.
- All serial communication interrupts at ring 0 in 32 bit mode.
- Uses transmit & receive FIFOs(16450, 16550, 16650, etc.).
- Win32, Win16, Console, & MS-DOS apps active concurrently.
- Over 400K baud under Windows 95 & Windows 3.x!!!
- Integrates seamlessly with all COMM-DRV products.
- Compatible with Visual C/C++, Visual Basic, and many more tools.

**Features Supported ByAll Products**

- Remap/Change baud rates/divisor.
- Product Customization.
- Hardware/Software Flow Control.
- Unlimited number of ports active concurrently.
- 8250/16450/16550 Auto detection/Up to 400 KBaud.
- 100% Port re-entrant code(Important for multitasking).
- Support all standard dumb multiport cards.

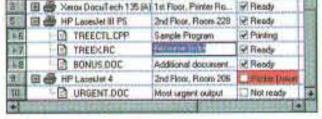
**WCSC Price List**

```
COMM-DRV/Lib(DOS,Win3x,Win95,NT)...$189.95
COMM-DRV/Lib(Competitive Upgrade)..$ 99.95
COMM-DRV/VxD.......................$189.95
COMM-DRV/Dos.......................$ 99.95
4Port Card(16550/8 bits)...........$170.00
4Port Card(16550/16 bits)..........$199.95
8Port Card(16550/16 bits)..........$225.00
8Port Card(16550/16 bits/Ext.Box)..$299.95
Professional Combo(Lib,VxD)........$329.95
Software Combo(Lib,VxD, Dos).......$399.95
Complete Combo(Sftwe+4Port(16550)..$499.95
```

**WCSC**
2470 S Dairy Ashford Suite 188, Houston, TX 77077
**1-800-966-4832 713-498-4832**
(Fax) 713-568-3334 (BBS) 713-568-6401
**Internet:** sales@wcscnet.com
Visa/Mastercard/Discover/AMEX/Checks/Approved P.O. **WWW:** http://www.wcscnet.com/home.htm

CIRCLE NO. 602 ON READER SERVICE CARD

## Windows OCX and DLL Controls



| | Printer / Files | Description | Ready |
|---|---|---|---|
| 3 | Xerox DocuTech 135 (A) | 1st Floor, Printer Ro... | ✓ Ready |
| 5 | HP LaserJet III PS | 2nd Floor, Room 229 | ✓ Ready |
| 6 | TREECTL.CPP | Sample Program | ✓ Printing |
| 7 | TREEX.RC | [illegible] | ✓ Ready |
| 8 | BONUS.DOC | Additional document... | ✓ Ready |
| 9 | HP LaserJet 4 | 2nd Floor, Room 206 | Printer Down |
| 10 | URGENT.DOC | Most urgent output | Not ready |

### SftTree Tree Control

From simple listbox with bitmaps to multi-line, multi-column, hierarchical data display, SftTree delivers! This 16-bit and 32-bit OCX control supports most popular environments including VB, Delphi, Access, etc. Also available as a DLL-based control for C and C++.

### SftTabs Tab Control

This DLL-based custom control supports C, C++ (MFC, OWL), Windows 3.1, Windows NT and Windows 95 with over 50 different tab styles!

Add tab controls to your 16-bit and 32-bit applications with this easy to use tab control!

softel vdm
NOT THE INDUSTRY STANDARD
11 Michigan Ave
Wharton, NJ 07885

***Call today for your free demo!***

(201) 366-9618
FAX (201) 366-3984
**http://www.softelvdm.com**

**Hipparchus™ GIS Enabling Technology**

Over 250 functions provide C or C++ developers with breakthrough GIS capabilities. Interface with *any* GUI and *any* DBMS. Ellipsoidal vector algebra enables seamless global coverage, lightning-fast spatial indexing, polygon overlay and more!

**New! Release 3.0 Libraries** for all 32-bit Intel platforms: DOS, Windows 95, NT, OS/2, Linux ($775). RISC Unix: Sun, IBM, DEC, HP, SGI and Mac ($1475). Includes Application Prototyper, Tutorial, Reference and Georama extensible world Atlas on CD-ROM (Windows 95).

**Geodyssey Limited**
500, 815 Eighth Avenue SW,
Calgary, Alberta, Canada T2P 3P2
403-234-9848 Fax 403-266-7117 http://www.geodyssey.com

CIRCLE NO. 603 ON READER SERVICE CARD

P R O D U C T D E B U T

**High Quality Software Protection since 1989**

- ✓ Secure against systematic "crackers"!
- ✓ WIBU®-BOX for LPT, COM and ADB, USB, transparent and cascadable, as PCCard and (E)ISA card.
- ✓ DOS, Windows™, Windows™ 95, Windows™ NT, MacOS™, OS/2®, Novell.
- ✓ Protection without source code modification
- ✓ CD-ROM Protection.
- ✓ API independent of programming language and operating system

**NEW:**

**WIBU-KEY CD-ROM**
**Software, online documentation and multimedia introduction.**

**WkLAN: network protection.**

WIBU SYSTEMS

**WIBU-SYSTEMS AG**
Rueppurrer Strasse 54
D-76137 Karlsruhe, Germany
Tel. +49-721-93172-0 · FAX 93172-22
CIS 100142,1674 · email: info@wibu.de

GRIFFIN TECHNOLOGIES INC
1617 St. Andrew Dr., Lawrence, KS 66047
Tel.(913)832-2070 · FAX (913) 832-8787
CIS 71141,3624 · email: sales@griftech.com

CIRCLE NO. 604 ON READER SERVICE CARD

✓Port I/O
✓DPRAM
✓Interrupts

**DriverX**™
**Hardware Control Lib/OCX**

**Simple:** No DDK development. Easily create and debug custom hardware control using VC++, VB4, or Delphi.

**Fast:** New Version 3 includes Kernel BASIC interpreter for running loops and ISRs entirely at kernel-mode for optimum performance.

**DFC**™ Create high-performance VxD, NT, and WDM kernel-mode drivers from common source.

**Driver Framework C++ Class Library**

The DFC library simplifies DDK development by providing base class implementations of many common driver tasks.

**Does your application edit source files, macros or scripts?**

**SourceView**™

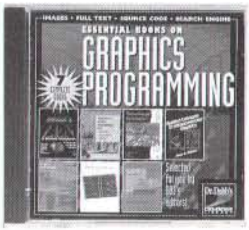
Add a syntax-highlighting text editor to your application with the SourceView editor library. Easily customized for any syntax. Examples provided for C, BASIC, and HTML.

**Tetradyne** Software Inc. **www.tetradyne.com**

(408) 377-6367 FAX: (408) 377-6258 tetradyne@mindspring.com

CIRCLE NO. 605 ON READER SERVICE CARD

FAST Search Engine



*Dr. Dobb's CD-ROM Library's*

## *Essential Books on Graphics Programming CD-ROM*

• 7 of the most important books ever written on Graphics Programming

Need to get up to speed on graphics programming—fast? There's only one definitive source you can turn to—*Dr. Dobb's Essential Books on Graphics Programming CD-ROM*!

Get seven of the most-essential books on graphics programming, with full text, diagrams, graphics, and source code—all on one CD-ROM. From fundamental algorithms to the most complex techniques, this CD-ROM lets you find all the critical information you need for your graphics programming projects.

**DDJ Editors' choices:**

- *Zen of Graphics Programming,* by Michael Abrash
- *Practical Image Processing in C,* by Craig A. Lindley
- *Photorealism and Ray Tracing in C,* by Watkins, Coy, and Finlay
- *Applied Concepts in Microcomputer Graphics,* by Bruce Artwick
- *Digital Halftoning,* by Robert Ulichney
- *Digital Image Warping,* by George Wolberg
- *Algorithms for Graphics and Image Processing,* by Theo Pavlidis

Texture mapping, color modeling, and morphing—all optimized for speedy 2-D and3-D graphics programming. Practical image acquisition and processing. High quality 3-D photorealistic graphics.

Digital halftoning. Image synthesis and special effects. And much more! Plus, all source code supplied with the books is included.

**Price: $69.95**

**Order Today!**
**U.S. & Canada:**
**800-992-0549**
**All Other Countries:**
**913-841-1631**
**E-mail:**
**orders@mfi.com**
**Fax Orders:**
**913-841-2624**

**Mail Orders**
*Dr. Dobb's CD-ROM Library*
1601 West 23rd St., Ste. 200
Lawrence, KS 66046-2700
USA

# DR. DOBB'S JOURNAL MARKETPLACE

## ASSEMBLY

**Pentium® Processor Optimizer**

Our software automatically annotates your assembly source with Pentium® processor optimization information (simulated pipeline pairing, AGI's, prefix stalls, register trace and more). Reports statistics by loop nesting depth. ($199.95) ASMFLOW generates flow charts, free diagrams, register analysis, x-ref, timing info and more—for 80x86, 8051, 8096, Z-80, 68HC11, 68HC05, and more. ($199.95) Call for free demo and catalog of assembly, TSR and floating point libraries. ($99.95 to $299.95)

Pentium is a registered trademark of Intel.

**Quantasm Corp.**
19672 Stevens Creek Blvd #307-D
Cupertino, CA 95014
800-765-8086 or 408-244-6826
FAX: 408-244-7268

CIRCLE NO. 501 ON READER SERVICE CARD

## COPYRIGHT

**InCopyright™** *Protect your creative work!*

**Copyright Your Software**

Sure, you can put the © symbol on your software and documentation, but for full legal protection, you must formally register your work with the U.S. Copyright Office. InCopyright is an easy, step-by-step Windows program that collects req'd. info and prints completed forms to your laser/inkjet printer. Saves time, money and maybe your job! Fully approved. Only $99 + $3 s&h.

**Parallel Resource, Inc.**
**PO Box 2488-DD • Auburn, AL 36831**
**(334)821-9000 http://www.parallelresource.com**

CIRCLE NO. 502 ON READER SERVICE CARD

## C++ LIBRARIES

**WMTLIB C++ Multithread Library**

Make multithread and multiprocess development simpler with the WMTLIB C++ classes, and gain cross-platform portability at the same time.

WMTLIB includes classes to manage synchronisation and data transfer between threads, both intraprocess and interprocess, with documented source, royalty free deployment, and inexpensive licensing per developer.

Supports Win 32 (NT and Win95), OS/2, LinuxThreads, Solaris 2.5 (Solaris threads and POSIX 1003.1c), UnixWare 2.1, more to follow.

See **www.westgold.com for more details.**

**Westongold Ltd.**

email: **info@westongold.com or call (+44) 1992 633801**

CIRCLE NO. 503 ON READER SERVICE CARD

## C++ TOOLS

Introducing **Code Navigator** for C++

This visual development environment takes away programming tedium. Program with built-in code patterns or build your own. Create your own CASE tools or custom documentation.

**Quintessoft Engineering, Inc.**
http://www.quintessoft.com (313) 669-0192

CIRCLE NO. 504 ON READER SERVICE CARD

## DEVELOPMENT TOOLS

**PARSING JUST GOT EASIER**

Parsing input used to mean tricky logic with lots of places for bugs to hide, slipped schedules, difficult maintenance. Nobody's idea of fun.

Now, more powerful, more convenient, ***AnaGram V1.5*** changes all that. You can take charge and concentrate on the real problems. Expressions? Database queries? Scripts? Command languages? No problem. *Call for free trial copy.*

***Anagram***™ by *Parsifal Software*
P.O. Box 219, Wayland, MA 01778
http://www.parsifalsoft.com
CIS: 72603,1763 info@parsifalsoft.com
(800) 879-2577 Voice/Fax (508) 358-2564

CIRCLE NO. 505 ON READER SERVICE CARD

**SPELL CHECKING & THESAURUS**
**MULTI-LINGUAL**

*Danish Dutch French German Italian*
*Norwegian Spanish Swedish UK English US English*

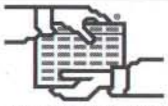
■ FIRST CLASS ■ AFFORDABLE ■ ROYALTY-FREE
■ FAST – EASY – POWERFUL
■ U.S. ENGLISH THESAURUS TOOLKIT TOO!
AVAILABLE FOR DOS, WIN 3.1, WIN NT, WIN 95, OS/2

LEXSAURUS SOFTWARE, INC.
427-3 Amherst St. Suite 303, Nashua, NH 03063
(603) 672-5941 • 1-800-570-1984 • Fax: (603) 672-5934
E-mail: info@lexsaurus.mv.com
**DOWNLOAD OUR FREE DEMO !**
**http://www.mv.com/ipusers/lexsaurus**

CIRCLE NO. 506 ON READER SERVICE CARD

NEW VERSION **Dr. DeeBee®** **ODBC Driver Kit**

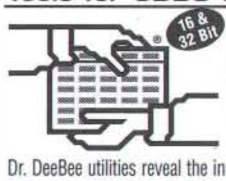
*The easiest way to create an ODBC driver*



Connect your proprietary database to the world's most popular database tools such as Access, Visual Basic, PowerBuilder, and Crystal Reports.

**Dr. DeeBee® ODBC Tools**

*Tools for ODBC development*



16 & 32 Bit

Ever wonder why your ODBC app is not working? Why it's just too slow? If the ODBC driver is OK?

Dr. DeeBee utilities reveal the inner workings of ODBC.

*SYWARE*®
P.O. Box 91 Kendall
Cambridge, MA 02142
**617-497-1376**
Fax 617-497-8729
http://www.syware.com

CIRCLE NO. 507 ON READER SERVICE CARD

## DEVELOPMENT TOOLS

**The McCabe OO ToolSet™**

The McCabe Object-Oriented ToolSet™ is the newest addition to the McCabe ToolSet™. It provides a wide variety of system metrics and a clear visualization of OO applications architecture. C++ systems can be fully tested and SAFE and UNSAFE classes identified. To reduce redundancy, the tool finds reusable code throughout a system. In addition, it contains features that help migrate systems from traditional languages to OO. The McCabe ToolSet is also available for over 25 traditional languages and dialects.

**Uncover Your Software** McCabe & Associates®

**For Further Information Call 800-638-6316 or Fax 410-995-1528**
**http://www.mccabe.com**

CIRCLE NO. 508 ON READER SERVICE CARD

**Btrieve Database Access Fast & Easy!**

Smithware — Controls for Btrieve

Get blinding performance and native access to Btrieve databases without writing code! Our advanced data binding technology outperforms the VB data control, ODBC, Jet and other access methods. Smithware's ActiveX Controls give you total control over your data with seamless integration of Btrieve's client/server extended operations, extended joined record sets, bound controls and more. Use our controls in your favorite development including VB, Delphi, VC++, Internet Explorer or in any environment that supports ActiveX. You can even enable your application over the web. Check out our benchmarks and Internet demo at **www.smithware.com/activex** or call us today and see for yourself how you can get fast easy access to your Btrieve Data!

***Smithware, Inc.***
2416 Hillsboro Road, Suite 201, Nashville, TN 37212
Phone: 615-386-3100 or 800-837-6316
E-mail: info@smithware.com
http://www.smithware.com

CIRCLE NO. 509 ON READER SERVICE CARD

## DEVICE DRIVERS

**Custom driver development for Windows 3.1, 95, NT, & OS/2.**

- **Kernel-mode device drivers, VxD's**
- **Printer & Graphics drivers**
- **Communications & Network solutions**

**Stratos Technology, Inc.**
**31 Schanck Road, Suite D**
**Freehold, NJ 07728**
**908-780-5220**
**Fax: 908-780-5302**

CIRCLE NO. 510 ON READER SERVICE CARD

**Custom Device Drivers - Consulting -**

| | |
|---|---|
| Operating Systems: | **Windows 95, 3.x, NT DOS, Embedded** |
| *Call us today for a quote!* | ***Rapid Development*** ***High Speed Execution*** ***Custom Hardware*** |

**Device Drivers International, LLC**

| **Sherman R. Couch** | **Anthony A. Kempka** |
|---|---|
| Tel: 513-777-8895 | Tel: 307-721-4881 |
| Fax: 513-723-5780 | Fax: 307-721-8788 |

CIRCLE NO. 511 ON READER SERVICE CARD

MARKETPLACE

## DEVICE DRIVERS

Boost your Basic!

Develop your app faster: Get TeraTech tools! 36 VB, QB, PB, C & Xbase libraries available. Call us and we'll mail you a free demo & booklet ASAP. Call now!

60 Day MBG! FREE demo!

* **Dazzle/VB** - images
* **VBlite** - print/comm
* **ProMath/VB** - numerics
* **FinLib/VB** - financial
* **VoxLib/Pro** - telephony
* **SpellCheck/VB** - spelling
* **ProBas** - & other 15 DOS libs
* **VB training** - 2 day intro class
* **Custom** - C, ASM, VB programming

**800-447-9120 ext.1275**

Dept 1275, 100 Park Avenue, Suite 360, Rockville MD 20850
Int'l:+1-301-424-3903 Fax:301-762-8185 BBS:301-762-8184
info@teratech.com http://www.teratech.com/teratech/

TeraTech tools for programmers™

CIRCLE NO. 512 ON READER SERVICE CARD

## EDITOR

**Win-Emacs™**
The Complete XEmacs Editor
for Microsoft Windows®

**NEW! 32 -bit version**
**for NT/Win95/Win 3.1**

**To Order Call 1-800-WIN-EMACS**

**Pearl Software Corporation**
Tel: 510-652-4361 • Fax: 510-652-4362
http://www.pearlsoft.com/

CIRCLE NO. 513 ON READER SERVICE CARD

## EDUCATION

**B.S. & M.S.**
In Computer Science

- ALL COURSES HOME STUDY
- Approved for tuition reimbursement by leading corporations
- Increase your earning power
- C++, Using Windows, and Ada courses available
- Career*Path* training in computer programming available

**For free information call:**
**1-800-767-AICS**

AMERICAN INSTITUTE FOR COMPUTER SCIENCES
ACCREDITED MEMBER
*World Association of Universities and Colleges*

## ENCRYPTION

carrick™

When you need to provide industrial strength encryption within your software projects, simply wire in carrick.

1 800 ENCRYPT
www.encryption.com

CIRCLE NO. 514 ON READER SERVICE CARD

## FORTRAN

**The SPINDRIFT Library Version 3.0**

*The Complete FORTRAN Programmer's Toolkit.*

Create sophisticated state-of-the-art user interfaces for your FORTRAN programs. Spindrift's callable subroutines and functions let you
- Have data entry screens, dialog boxes, scrolling list boxes, pull-down menus, push buttons, help panels, etc. All with full mouse support.
- Much more! 281 subroutines and functions for keyboard, screen, mouse, DOS control, and hardware status.

Comprehensive demonstration program shows what you can do and how to do it.
*Now shipping Version 3.0 with all new User Guide.*

**PRICE: 16 Bit Compilers** - $149 **32 Bit Compilers** - $289

**Spindrift Laboratories, Ltd.**
8 Tanglewood Path • Galena • IL 61036 • USA
Phone: (815) 777-8240 • FAX: (815) 777-8241

CIRCLE NO. 515 ON READER SERVICE CARD

## GRAPHIC LIBRARIES

★★★★★
VICTOR
Image Processing Library

Fast **BMP, TIFF, PCX, GIF, TGA, JPEG**. Adjust brightness, contrast, sharpen, create filters, resize, rotate, +more of single image, multiple images, or any image area; color reduction to optimum, specific, or std. palette; print; scan; crop, combine, compare, blend images. EGA, VGA, SVGA.

DOS $199, 16-bit DLL $299, 32-bit DLL $499

Catenary Systems
314-962-7833/fax: 314-962-8037
www.catenary.com/victor
ask for free demo src avail visa/mc/c.o.d.

CIRCLE NO. 516 ON READER SERVICE CARD

## HOME AUTOMATION

**HOMATIONS SYSTEMS**
**Graphic Home Automation for Windows 95**

**HomationBasic - $189.95**
- Includes serial port X–10 Interface
- Easy to use graphic software
- All 256 X-10 addresses
- Easy to use timers and events

**Homation 2000 - $295**
- Multiple floor plan layouts
- OLE Automation server
- Digital thermostats – energy management

http://www.homation.com • 800-522-6054

CIRCLE NO. 517 ON READER SERVICE CARD

## IMAGING LIBRARIES

CORPORATE SUITE IMAGELIB FOR DELPHI/VB INTERFACE

*"The most robust document imaging tool in today's market."*

free trial download on: http://www.ImageLib.com
CALL (800) 404-3832
Skyline Tools
Windows Stellar JULY 1996

CIRCLE NO. 518 ON READER SERVICE CARD

## LIBRARIES

**Complete C++ Math Library**
- **Complete multidimensional array language**
- **Complete package of mathematicc**
- **Full error handling package**

**Advanced Support Modules**

| | |
|---|---|
| **Signal Processing** *(NEW)* | **Finite Element** |
| **Financial Utility***(NEW)* | **Least Squares** |

**Call for complete list**

***FREE* demo program on website**

http://www.wolfenet.com/~dyad dyad@wolfenet.com
**Dyad Software Corporation**
6947 Coal Ck Pkwy SE #361 Newcastle WA 98059
(206) 637-9426 (800) 366-1563

CIRCLE NO. 519 ON READER SERVICE CARD

## MIDDLEWARE

**NEW! App-Link RADX Control Set**
Rapid Application Data eXchange

Vis C++ · VB · MS IE · Excel · FoxPro · Your App · Container X · Delphi · Optima · Access

The new App-Link RADX control set includes a VBX and 16/32 bit ActiveX (OCX/OLE) controls. Use RADX to easily send and receive messages between any apps that can support one of the RADX control types. Messages can be sent between 16 and 32 bit platforms, locally or over a LAN! RADX is VERY easy to use, fast, and reliable. These are high quality messaging controls that every Windows programmer should own. Features include queuing, prioritization, auto app load, event driven msg receipt, and much more! Forget the hassles of DDE, NETDDE and OLE - Use App-Link RADX! RADX is only $149.95, and is royalty free!

**Please visit our WWW site for details:**
**http://www.synergysw.com**

Synergy Software Technologies Inc.
159 Pearl Street
Essex Junction, Vermont 05452
sales: 800-294-8514, tel: 802-878-8514
fax: 802-878-4055
sst@synergysw.com

CIRCLE NO. 520 ON READER SERVICE CARD

## NEW SCAN TOOLS

DLL - OCX - VBX
- **Barcode recognition from bitmap**
- **Barcode + text write into bitmap**
- **Skew correction** • **Noise + Crop**
- **Annotate image** • **Line removal**
- **TIFF r/w - de/compress - fast scale**

**Axtel, Inc.** Tel: (714) 964-6666 Fax: (714) 964-6766

Free demo on http://www.axtel.com

CIRCLE NO. 521 ON READER SERVICE CARD

## PROLOG TOOLS

Add Intelligent Components to C/C++, VB, Delphi & the Web

*Recommend products, diagnose problems, give advice, use Java, CGI and Sockets on the Web, access ODBC data. Amzi!™ Logic Server™ tools and libraries (DLLs) for Win NT 95 3.x, DOS & Solaris. FREEWARE & DEMOS on our Web Site.*

**Amzi! inc.**
40 Samuel Prescott
Stow, MA 01775 USA

***Visit*** www.amzi.com
***Email*** info@amzi.com
***Call*** +1-508-897-7332
***Fax*** +1-508-897-2784

## SECURITY

Rodney's
**Copy Guard**
**Delphi Software Protection**
- True Native Delphi VCL wrapper, not a DLL
- Used fixed or floating licensing scheme to protect your network application.
- Fully event driven
- Does not use a Hardware Key or Disk Key

**It's So Easy to:**
- Add Password Protection
- Set applications to expire after 30-90 days
- Create fully/partially functional demo copies
- Setup full Version or Demo mode
- Prohibit copying to unauthorized computers
- Specify the number of concurrent users that can run your application on a network
- Set applications to expire after a given date or given number of executions

***PLUS MUCH MORE:

One Time Fee:
16 bit .... $159
32 bit .... $159
Combo Pack (Both 16 and 32 bit) ....$199

Call: 800-477-6163 or 619-669-6971
E-mail: sales@marotz.com
Fax: 619-669-6914

Marotz, Inc.
**60 day Guarantee** www.marotz.com

CIRCLE NO. 522 ON READER SERVICE CARD

**Lock Out Pirates!**
**Lock In Profits!**

The Marx CRYPTO-BOX is the result of over 10 years experience in **effective software protection.**
- microprocessor controls ID codes, memory, dynamic algorithm and high speed data encryption
- remote access to passwords and counters
- license metering in networks: single key per LAN

404-321-3020 FAX: 404-321-0760
**1-800-MARX INT**
MARX INTERNATIONAL, INC.
20 Executive Park West, Suite 2027, Atlanta, GA 30329
Visit our Home Page: http://www.marx.com

CIRCLE NO. 523 ON READER SERVICE CARD

MARKETPLACE

## SECURITY

**Az-Tech Software, Inc.**
**Leaders in Software Security**

**EVERLOCK**
(Software Based)
**SAFE-D**
(Secure Distribution)
**EVERKEY**
(Hardware Based)

INTERNET
CPU-LOCK
HDD-LOCK
CD-ROM LOCK
REMOTE ACTIVATION
EXECUTION LIMITS
LIMITED INSTALLS

REMOTE RESETS
SERIALIZATION
REGISTRATION
ACCESS FLAGS
DATE LIMITS
NET LIMITS
"POINT & SHOOT"
WIN95*3.11*DOS

Phone: (816) 776-2700 x325
Fax: (816) 776-8398 * Dept: SMG-325
Internet: www.az-tech.com
E-Mail: sales@az-tech.com

CIRCLE NO. 524 ON READER SERVICE CARD

## SECURITY

**CrypKey Software Licensing System**
"Software protection with **NO** hardware lock and **NO** disk key "

CrypKey is **software copy protection** that is:
* completely secure from any disk copy program
* perfect for **CD-ROM** or **INTERNET** distribution!
* cost effective, user friendly, and **100% guaranteed** to satisfy!

CrypKey can **increase your software sales:**
* upsell **options and levels** of your software
* **lease or demo** your software by runs or time
* **enable or upgrade** your customers **instantly** by **phone, fax or email!**

**New!** unique **Ready-To-Try** features upon install allows 1 trial period only per customer.
**New!** unique **Add-On** feature-add more options, levels, runs or time to existing licenses.
**New! CrypKey Instant**-protects in just **5 minutes** with **no source code changes.**

CrypKey is completely compatible with MS-DOS, MS-Windows 3.x, Win32s, Win95, Win NT, and manages network licenses on all Novell and Microsoft operating system based networks. CrypKey is produced by Kenonic Controls Ltd.—engineering and software since 1972.

**Kenonic Controls Limited**
**7175 - 12th Street South East**
**Calgary, Alberta, Canada T2H 2S6**
**(403) 258-6200 fax: (403) 258-6201**
**INTERNET: crypkey@kenonic.com**
**WEB: http://www.kenonic.com/crypkey.html**

CIRCLE NO. 526 ON READER SERVICE CARD

## TOOLS

**C and C++ DOCUMENTATION TOOLS (v. 6.2)**

* **C-CALL ($69)** Graphic-tree of caller/called function hierarchy, cross-reference, file/function index.
* **C-CMT ($69)** Creates/inserts/updates comment-blocks (functions/identifiers used) for each function.
* **C-METRIC ($59)** Calculates path complexity, counts lines with comments, code, 'C' statements.
* **C-LIST ($69)** Lists and action-diagrams, or reformats source into user-selected standard formats.
* **C-REF ($69)** Creates cross-reference of local/global/define/parameter identifiers.
* **C-DOC ($199) PACKAGE** All 5 programs integrated as DOS program. <10,000 lines. C-BROWSE Windows graphic-tree viewer.
* **C-DOC Professional ($299)** DOS, Windows, OS/2. 3-ring binder/case. Processes 1,000,000+ lines.

**SOFTWARE BLACKSMITHS INC.**
6064 St Ives Way, Mississauga
ONT Canada L5N-4M1
email @ swbs.com
Voice/Fax **(905) 858-4466**
http://www.swbs.com

CIRCLE NO. 528 ON READER SERVICE CARD

## LIBRARIES

**Add *ZIP* (and *UNZIP* too!) to *your* Windows applications!**

**DynaZIP™** *3.0* Data Compression Toolkits

The new ROYALTY-FREE DynaZIP family of developer's tools let you add ZIP and UNZIP capabilities to your Windows applications. No more "shelling" to DOS, no more fussing with proprietary compression formats. DLLs, VBXs, OCXs and a new database interface provide full access from many languages. Fast, reliable, and easy to use! 16 and 32 bit versions, supports long filenames.

*Fully supported, 30-day no-risk guarantee!*
*Call today, toll free: (800) 962-2949*

Inner Media, Inc., Hollis, NH USA (603) 465-3216, fax (603) 465-7195
Visit our web site at http://www.innermedia.com

CIRCLE NO. 525 ON READER SERVICE CARD

## TRAINING

**GORDIAN**
INSTITUTE
*Advanced Software Technology Training*

**Making Sense of Data:**
**Computer-Aided Pattern Discovery**
*January 27-31, 1997 • Registration $1995*

*To receive course details and a curriculum listing of other Gordian Institute offerings, provide a fax number or Email address to:*

800-405-2114
gordian@heuristics.com
www.heuristics.com/gordian

The Gordian Institute is a division of American Heuristics Corporation
www.heuristics.com • 304-547-4201 • info@heuristics.com

CIRCLE NO. 527 ON READER SERVICE CARD

**Powerful Protection, Affordable Price**
**Software Security's UniKey® Anti-Piracy System**

* Reliable hardware key
* Secure protection for multiple applications/versions
* The industry's only patented protection technology
* Protects applications quickly, without programming
* Remote update capabilities
* DOS/Windows/95/NT/Unix/ NetWare support
* Time/date-based execution control
* Made in U.S.A.

**Call now and order your UniKey Developer's Kit:**
**800-841-1316**

SOFTWARE **SECURITY**
6 Thorndal Circle, Darien, CT
06820-5421 (203)656-3000
http://www.softsec.com

CIRCLE NO. 529 ON READER SERVICE CARD

# DR. DOBB'S WEB DIRECTORY

http://www.DDJ.com

## CASE/TESTING & DESIGN

**PURE ATRIA**
**http://www.pureatria.com**

## CD ROM

**INFOMAGIC**
**http://www.infomagic.com**

## CONSULTATION/SYSTEMS INTEGRATOR

**UNIPLAN**
**http://www.uniplan.it**

## CROSS PLATFORM/GUI DEV. TOOLS

**SOFTEL VDM**
**http://www.softelvdm.com**

**GAMELON by MENAI**
**http://www.menai.com/**

## DOCUMENTATION TOOLS

**Willows Software**
**http://www.willows.com**
**Use The Willows Toolkit for porting 16- and 32-bit Windows™ applications to UNIX, Macintosh, real-time, network and NC operating systems, JAVA, and other platforms.**

## EMBEDDED SYSTEMS

**WIND RIVER SYSTEMS**
**http://www.wrs.com**

## LANGUAGES/DEVELOPMENT TOOLS

**INTEL**
**http://www.intel.com/ial/vtune/index/html**

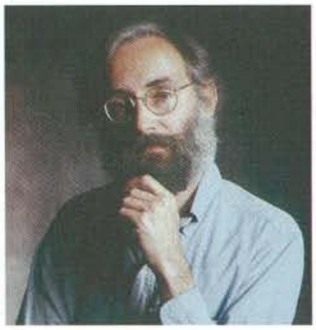
**INTERACTIVE SOFTWARE ENGINEERING**
**http://www.eiffel.com**

## MISCELLANEOUS

**SYMANTEC**
**http://www.careermosaic.com/cm/symantec/home.html**

## UTILITIES

**NuMEGA**
**http://www.numega.com/**



# *Is Thin In?*

Mo asked Joe, "So what are you grinning about?" Joe Weaver was indeed grinning over his cream soda when Maureen "Mo" McBean walked into Foo Bar that night. British journalist Larry Wilde and I had each tried to find out what Joe was so tickled about, but he was apparently waiting for a full house. Now the place was about as full as it gets, with three very regular customers and me, the relief bartender.

"Oh, nothing," Joe answered. "Well, something, I guess." He was obviously bursting with pride. "One of my leads got picked up for a cover blurb."

"So some of your deathless prose appeared on the cover of that barf bag you write for?"

"It's called *Words on the Wing*," Joe sniffed. "It's an in-flight magazine."

"That's what I meant. Give me a Leinenkugel, Mike. It's like that old joke about Niklaus Wirth, the inventor of Pascal. In Europe they pronounce it 'nee-claus veert', but in America they pronounce it 'nickle's worth.' It's the difference between call-by-name and call-by-value." Joe had the look of a surfer who, riding a big wave a second before, has just caught his board in the back of the head. Larry threw him a line.

"What was the lead, Joseph? I dare say it was something clever if they picked it up for the cover."

"'Don't know paperback from pay-per-view?'" Joe said, in that pompous sing-song voice writers always use when quoting themselves, "'Check out our massive media review.'"

Mo snorted. "That was your lead? It sounds like ad copy!"

There was a moment of awkward silence. I wiped the bar, chipping away with my thumbnail at a chunk of bar grit that I suspected may have been a peanut in happier times.

Mo gave Joe a sidelong look, sighed, and gave in. "Cute, though," she said, hating herself for doing it.

"Do you think so?" Joe bubbled. It's hard to keep anyone as carbonated as Joe down for long. "Cute is what I was going for. It's the hallmark of my style," he added, making it sound like another self quote.

"Yeah, well," Mo conceded grudgingly, "style matters. I guess."

"Style," Larry proclaimed, "is all. It is the sole end of all art, the true message of most speech, and the one quality that makes the humblest Briton superior to the entire Royal Family."

"Well, I write about the computer industry," Mo said. "Wouldn't you say style is less important there?"

"I would not, and I can give you an example. Last year, Oracle's Larry Ellison gave a bombastic speech telling developers, essentially, that they'd better jump on the Network Computer bandwagon before the NC crushed the PC into dust. Did his brash style alienate developers? It's possible. Style matters."

"Oracle. Isn't their net address oracle@delphi.com?"

"You make my point for me. Ellison's style invites just that sort of disparaging gag, however lame."

"Well, we all knew he was not PC."

"Please! But the whole future of these NCs, these thin clients, hangs on a style question: Is thin in?"

"Speaking of thin computers," Joe cut in, "have you seen Apple's Shay? It's a Newton with a keyboard in a clamshell case that Apple plans to sell as a laptop."

"Beastly name, 'Shay,'" Larry said, sipping thoughtfully at his white wine.

"Still, I suppose they could hardly call it the Clamshell Newt, newts being amphibians rather than crustaceans."

"Newt Gingrich is a crustacean."

"You may have something there. Doubtless you caught it from Maureen."

"Anyhoo, you have to hand it to Apple," Joe said, "the thing sure is cute."

Mo drained her glass and clunked it on the bar. "Cute. Right. I believe that's the hallmark of their style."

Michael Swaine

Michael Swaine
editor-at-large
mswaine@cruzio.com

# Now discover CodeBase 6—The fastest and smallest database engine available with xBASE file compatibility!

Query a million records in just 0.97 of a second, or add ten thousand records in just 2.17 seconds.

Discover these valuable benefits of CodeBase:

- C, C++, Visual Basic, Delphi and Java are all supported
- FoxPro, Clipper and dBASE file support, including full multi-user compatibility
- Runs under Windows 95, NT, 3.1, DOS, Mac, OS/2, Solaris, SunOS, AIX, SCO, Linux, UnixWare...
- Tiny libraries help apps load quickly and use less memory
- 100% royalty-free client/server configuration included
- Transaction processing and logging
- Crash recovery is built-in
- Full featured report writer included
- Source code included
- Royalty free distribution

**New ActiveX**

Seamlessly integrate our fast database engine with your favorite compiler using our new ActiveX database support.

1991 DATA BASED ADVISOR Readers Choice Award
1992 DATA BASED ADVISOR Readers Choice Award
1993 DATA BASED ADVISOR Readers Choice Award
1994 DATA BASED ADVISOR Readers Choice Award
1995 DATA BASED ADVISOR Readers Choice Award

*Award-Winning performance 5 years in a row!*

"CodeBase gives ACT! the fast database access contact management users need"

- Michael Plasterer, Director of Development, Symantec

*The Database Engine for Programmers*

## FREE 30 Day Test Drive

Test drive the new CodeBase 6 for 30 days with your own code. No risk. No obligation. Order yours today.

Call: 403-437-2410

SEQUITER SOFTWARE INC.
P.O. Box 783 Greenland NH 03840
Fax: 403-436-2999
Email: info@sequiter.com
Web site: www.sequiter.com

CIRCLE NO. 105 ON READER SERVICE CARD

# Build front end user interfaces for C++ faster and more productively with Delphi 2.0

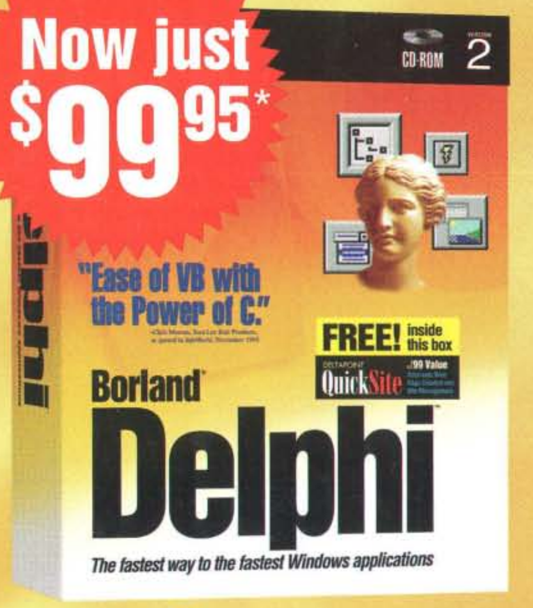
"It takes too long to write everything in C++, so we write the user interface in Delphi. It really propels us through application development. The DLLs to actually run the instruments are written in C++. This works very well for us."

—Doug Weigel, MTS, Hewlett-Packard

"Borland's Delphi 2.0 gives developers a language almost as readable as BASIC, but a whole lot faster and with genuine object orientation. It features performance competitive with comprehensive development environments such as Microsoft's Visual C++."

—David Berlind, *PCWeek* Labs, April 1996

**Now just $99^{95}$***


CD-ROM 2
"Ease of VB with the Power of C."
FREE! inside this box
QuickSite
Borland
Delphi
The fastest way to the fastest Windows applications

**To get up and running quickly, Delphi 2.0 includes:**

- ✓ Using Delphi with C++ Reference
- ✓ *FREE best-selling book* Teach Yourself Delphi 2 in 21 Days
- ✓ *Step-by-step online tutorials*
- ✓ *Easy-to-use visual tools*
- ✓ *More than 90 reusable components*
- ✓ *The world's fastest native-code compiler*
- ✓ *Full-featured Integrated Development Environment (IDE)*
- ✓ *Support for 16- and 32-bit development*

**New Delphi™ 2.0 is the perfect complement to C++**—it propels C++ programmers through application development. With Delphi 2.0, you can rapidly prototype a functional application shell and drastically reduce the time it takes to build a complete application interface. You can even easily host Delphi UI forms in MFC or OWL.

Unlike Visual Basic, Delphi offers C++ developers two major advantages: small, fast executables that run well on Windows 3.1, Windows 95, and Windows NT, and a unique combination of a native-code compiler and visual development that minimizes coding. Take advantage of drag-and-drop development with more than 90 reusable components right in the box, or easily create your own and add them to the Component Palette. Be productive with a full-featured IDE that includes integrated debugging, tracing, easy-to-understand error messages, and color syntax highlighting. Even use or customize ActiveX controls right in the Delphi environment with full 32-bit support for OLE automation. And, to get you started right away, Delphi 2.0 includes a step-by-step online reference designed exclusively for C++ developers that shows you how to link C++ code to a Delphi UI. Plus, you'll get the best-selling book, *Teach Yourself Delphi 2 in 21 Days*, absolutely FREE.

**For more information on Delphi 2.0, call 1-800-336-6464, ext. 50854, or check out Borland Online at http://www.borland.com/delphi99/**

Designed for Microsoft® BackOffice™

**Borland**
Making Development Easier

| | | | |
|---|---|---|---|
| **Best Buy** | **CompUSA** | **Computer City** | **MicroWarehouse** |
| 1-800-380-7278 | 1-800-COMPUSA | 1-800-THE-CITY | 1-800-367-7080 |

CIRCLE NO. 67 ON READER SERVICE CARD